UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

CENTRO DE INVESTIGACIÓN EN TECNOLOGÍAS DE INFORMACIÓN Y SISTEMAS

MAESTRÍA EN AUTOMATIZACIÓN Y CONTROL

DISEÑO DE UN SISTEMA INTELIGENTE EMBEBIDO PARA CONTROL DE UN SISTEMA SISO NO LINEAL

T E S I S

QUE PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN AUTOMATIZACIÓN Y CONTROL

PRESENTA:

ING. MARIO ALBERTO SALDAÑA ORTÍZ

ASESORES:

DR. LUIS ENRIQUE RAMOS VELASCO

DR. ABEL GARCÍA BARRIENTOS

PACHUCA HGO., MÉXICO 18 DE ENERO DE 2016

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO
Instituto de Ciencias Básicas e Ingeniería
Institute of Basic Sciences and Engineering
Área Académica de Computación y Electrónica
Computer Science and Electronics Department

Mineral de la Reforma, Hgo., a
Oficio No. MCAC03_2016

Ing. Mario Alberto Saldaña Ortiz
P R E S E N T E

Por medio de la presente y en mi calidad de coordinador de la Maestría en Ciencias en Automatización y Control, del Área Académica de Computación y Electrónica (AACyE) de la Universidad Autónoma del Estado de Hidalgo (UAEH), me es grato informarle que el Jurado asignado para la revisión de su trabajo de tesis titulado: "**Diseño de un sistema inteligente embebido para control de un sistema SISO no lineal**", dirigido por el Dr. Luis Enrique Ramos Velasco y el Dr. Abel García Barrientos, que para obtener el grado de Maestro en Ciencias en Automatización y Control fue presentado por usted, ha tenido a bien en reunión de sinodales, autorizarlo para impresión. A continuación se integran las firmas de conformidad de los integrantes del Jurado:

Dr. Joel Suárez Cansino	(Presidente)	AACyE-UAEH	
Dr. Carlos Cuvas Castillo	(Secretario)	CINVESTAV-IPN	Carlos Cuvas C.
Dr. Luis Enrique Ramos Velasco	(Vocal)	AACyE-UAEH	
Dr. Abel García Barrientos	(Suplente)	ITSM	

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

Atentamente
"Amor, Orden y Progreso"

Dr. Omar Jacobo Santes Sánchez
Coordinador de la Maestría en Ciencias en Automatización y Control
Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería
Área Académica de Computación y Electrónica

c.c.p. Dr. Orlando Ávila Pozos, Director del Instituto de Ciencias Básicas e Ingeniería
c.c.p. Dr. Hugo Romero Trejo, Jefe del Área Académica de Computación y Electrónica
c.c.p. Expediente/ apl

Ciudad del Conocimiento
Carretera Pachuca - Tulancingo km. 4.5
Colonia Carboneras
Mineral de la Reforma, Hidalgo, México, C.P. 42184
Tel. +52 771 7172000 exts. 2250 y 2251
omarj@uaeh.edu.mx

www.uaeh.edu.mx

Ing. Mario Alberto Saldaña Ortíz

Diseño de un Sistema Inteligente Embebido para Control de un Sistema SISO No Lineal

El presente trabajo, es fruto del esfuerzo de dos años,
de trabajo y dedicación
está dedicado a mis padres,
quienes me han educado y enseñado,
me han dado su cariño y comprensión,
me han dado los medios para concluir
de manera exitosa todos mis proyectos.
Por todo eso y mucho más, GRACIAS.

Gracias a

CONACYT por la beca de maestría otorgada durante el periodo Agosto 2013 - Julio 2015, con número de registro 295130.

CITIS - UAEH por la formación académica brindada a lo largo de mi estancia en sus instalaciones.

LSIS - Universite de Toulon por las facilidades y el apoyo académico brindado que me ha permitido realizar este trabajo de investigación a través de la beca mixta de CONACYT periodo octubre - diciembre 2015.

Agradecimientos

Agradezco a la Universidad Autónoma del Estado de Hidalgo, institución que me dio los medios para culminar de una manera exitosa la *Maestría en Automatización y Control* y de este modo crecer personal y profesionalmente.

A mis padres Mario Alberto y Rosa Olivia, por su amor incondicional, por ser un excelente ejemplo de sacrificio y esfuerzo para mí, ustedes me inspiran a ser mejor y me dan las fuerza necesaria para afrontar mis problemas con entusiasmo, me han enseñado a vivir, me han cuidado y educado, quienes incluso han dado todo por verme feliz, es por eso que para ustedes y por ustedes todo lo que soy.

Al Dr. Abel García Barrientos por todo su apoyo, consejos y tiempo que me brindo.

Al Dr. Luis Enrique Ramos Velasco, por todo el tiempo dedicado a desarrollar este trabajo de investigación, por su paciencia y por todo el conocimiento que me compartió, por toda esa motivación que jamas me dejo de brindar, por demostrar que en la vida siempre hay que gritar al mundo que se debe ser alegre.

A mis amigos y compañeros de esta gran aventura llamada MAESTRÍA, que siempre fueron un pilar importante para trabajar, gracias a ustedes que siempre de las formas más inesperadas hacían que un día tedioso, aburrido y estresante se convirtiera en uno agradable y lleno de risas, gracias por todo su apoyo e ideas aportadas durante todo este tiempo y, sobre todo, por su valiosa amistad.

Al Dr. Jean Francois Balman y al Dr. Frédéric Lafont, quienes aportaron muchas ideas y observaciones importantes, ambos mejoraron en gran manera mi forma de trabajar e investigar.

Al Dr. Omar Jacobo Santos Sánchez por todo su apoyo para concluir esta maestría, por ser para mi un investigador modelo, por enseñarme que nunca hay que dejar de lado nuestros valores y por que siempre, a su manera, logro motivarme para nunca desistir.

Finalmente, pero no menos importante, agradezco a todos los que de manera indirecta hicieron una contribución a este trabajo, gracias a sus estudios y publicaciones, a todos los hombres de ciencia y letras que de uno u otro modo han dado vida al

presente documento y que además han aportado algo a mi formación.

Resumen

Diseño de un Sistema Inteligente Embebido para Control de un Sistema SISO No Lineal

En la práctica de la ingeniería comúnmente existe la necesidad de llevar un sistema físico hasta un punto determinado con cierto grado de exactitud. Cada día son más las aplicaciones en las que se ven involucrados los sistemas de control, ya sean sistemas de transporte, militares, industriales, líneas de ensamble, sistemas para uso agrícola, sistemas de seguridad, producción industrial, etc. En algunos de éstos, la precisión y la rapidez son de vital importancia. Una alternativa para incrementar la precisión y la rapidez de los sistemas de control es la implementación de los sistemas electrónicos digitales, en las últimas décadas los sistemas electrónicos digitales han tenido un rápido desarrollo. En el presente trabajo de tesis se propone un esquema de control PID Wavenet Difuso para controlar la velocidad de un motor y la posición angular del carro - péndulo invertido. Dicho controlador sintoniza en línea las ganancias proporcional, integral y derivativa de un controlador PID discreto, mediante la identificación del sistema no-lineal de modelo matemático desconocido. Esto se logra empleando una red neuronal artificial de base radial con funciones de activación wavelet hijas (Wavenet) y un filtro de respuesta infinita al impulso (IIR) en cascada. Se realiza el diseño digital VLSI del controlador utilizando el lenguaje de descripción de hardware (HDL) VHDL a nivel de transferencia de registros (RTL). El algoritmo de control es programado en Simulink y se obtienen los siguientes resultados.

1. Simulación

    a) Control de velocidad de un motor de CD.

    b) Control de posición angular del carro - péndulo invertido.

    c) Análisis comparativo entre PID Wavenet, PID Wavenet-Difuso.

2. Experimental

    a) Control de velocidad de un motor de CD.

Abstract

Embedded Intelligent System Design for a Non-Linear SISO System Control

In the practice of engineering, so often, exist the necessity to take a physical system to a reference point with some grade of exactitude. Nowadays the applications where the control systems are involved are more, in transport systems, militaries, industrial, assembly lines, agriculture systems, security systems, industrial production, etc. In some of these cases, the accuracy and speed are fundamental and very important. An alternative to improve the accuracy and speed of the control systems is the implementation of the digital electronic systems, which, in the last decades, had have a huge development. In this thesis work, a PID Fuzzy Wavenet control scheme is proposed to control the speed of a DC motor and the angular position of an inverted pendulum - car. This controller tunes online the proportional, integral and derivative gains of discrete PID, through the identification of the nonlinear system with an unknown mathematical model. This is achieved using an artificial neural network of radial base with daughter wavelets activations functions (Wavenet) and one impulse infinite response filter (IIR) on cascade. The digital VLSI design is made using the hardware description language (HDL) VHDL at a register transfer level (RTL). With the finality of validate the VLSI architecture and the controller performance, the control algorithm is programmed on Simulink obtaining the next results.

1. Simulation

    a) Speed control of a DC motor.

    b) Angular position control of the inverted pendulum - car.

    c) Comparative analysis between the Wavenet PID and the Fuzzy - Wavenet PID.

2. Experimental

    a) Speed control of a DC motor.

Publicaciones

Como parte de los resultados del trabajo de investigación desarrollado en esta tesis, se obtuvieron los siguientes artículos:

Articulos en congresos internacionales.

Aceptados

- **M.A. Saldaña-O., A. García-Barrientos, L.E. Ramos-V, Jean-Fracois Balmat and Frédéric Lafont**,*A Comparative Study of the Wavenet PID Controllers for Applications in Non-Linear Systems*, Asia-Pacific Conference on Computer Aided System Engineering (APCASE) 2015, Quito, Ecuador, July 2015.

Articulos en congresos nacionales.

Aceptados

- **M.A. Saldaña, L.E. Ramos-V, J.P. Ordaz-O, A. García-Barrientos, I. Algredo-Badillo, Jean-Francois Balmat and Frédéric Lafont.**,*A Comparative Study of the Wavenet PID Controllers for Aplications in Non-Linear Systems*, 12th International Conference on Electrical Engineering, Computing Science and Automatic Control 2015, México, Octubre 2015.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En la práctica de la ingeniería de control comúnmente existe la necesidad de llevar un sistema físico hasta un punto de operación determinado con cierto grado de exactitud. Un sistema de control es el conjunto de técnicas y herramientas que guían a un sistema físico hasta las condiciones deseadas. Cada día, son más las aplicaciones en las que se ven involucrados los sistemas de control, ya sean sistemas de transporte, militares, industriales, líneas de ensamble, sistemas para uso agrícola, sistemas de seguridad, producción industrial, etc. En algunos de éstos, la precisión y la rapidez son de vital importancia. Una alternativa para incrementar la precisión y la rapidez de los sistemas de control es la implementación de los sistemas electrónicos digitales, en las últimas décadas los sistemas electrónicos digitales han tenido un rápido desarrollo. En particular, el número de transistores en un circuito integrado (IC, Integrated Circuit) ha crecido en forma increíble, por lo que hoy en día es posible encontrar un sinfín de aplicaciones para estos sistemas. Los ICs pueden ser clasificados en función al número de compuertas lógicas que contienen en su interior como en [**?**, **?**]:

- Entre 0 - 10 : Integración a pequeña escala (SSI, Small Scale Integration).
- Entre 11 - 100 : Integración a mediana escala (MSI, Medium Scale Integration).
- Entre 101 - 1000 : Integración a gran escala (LSI, Large Scale Integration).
- Más de 1000 : Integración a muy gran escala (VLSI, Very Large Scale Integration).

Actualmente la investigación y el desarrollo se enfoca en los ICs digitales *VLSI*, algunos ejemplos de estos son: los circuitos integrados de aplicación específica (ASIC, *Application Specific Integrated Circuits*), los procesadores digitales de señales (DSP, *Digital Signal Processor*), los dispositivos lógicos programables (PLD, *Programmable Logic Device*) y los arreglos de compuertas programables en campo (FPGA, *Field*

Programmable Gate Array). Este último tiene las ventajas de ser reconfigurable, permite el procesamiento en paralelo, tiene bajo consumo de potencia y es de bajo costo para el diseño de prototipos, por ello es el dispositivo ideal para este trabajo de tesis.

Otra tendencia importante en el diseño digital es el uso de lenguajes de descripción de hardware (HDL, Hardware Description Language), ya que permiten describir en forma textual circuitos digitales. También se utilizan junto con herramientas de síntesis lógica para automatizar el diseño. Los HDLs más utilizados actualmente por los diseñadores de lógica digital son Verilog y VHDL, ambos son lenguajes estándar de la IEEE para simulación y modelado de hardware [**?**, **?**, **?**].

Al igual que en los circuitos digitales, en los últimos años, se ha incrementado el estudio y aplicación de las Redes Neuronales Artificiales (RNA) en las áreas de computo inteligente y control, esto debido principalmente a su capacidad de aprendizaje e identificación de sistemas. También, la lógica difusa se emplea para transmitir la experiencia del humano para modelar y controlar sistemas dinámicos, estas dos características son muy útiles para el área de control, ya que estas son capaces de hacer la identificación de la planta a controlar sin necesidad de tener el modelo matemático de la misma y diseñarle un control bajo condiciones de incertidumbre.

El controlador más utilizado en la industria es el PID, debido a la facilidad con que se realiza e implementa, además de que presenta robustez [**?**]. Sin embargo, el algoritmo PID lineal es difícil de utilizar cuando el proceso a controlar presenta dinámicas complejas como zonas muertas y características altamente no lineales. El funcionamiento del controlador PID en general se basa en actuar de manera proporcional, integral y derivativa sobre la señal de error $e(t)$, definida como la diferencia entre la señal de referencia $y_{ref}(t)$ y la señal de salida del proceso $y(t)$, obteniendo así la señal de control $u(t)$, esta es inyectada como entrada al proceso para manipular la salida de éste en forma deseada, este controlador es ilustrado en la Figura **??**.

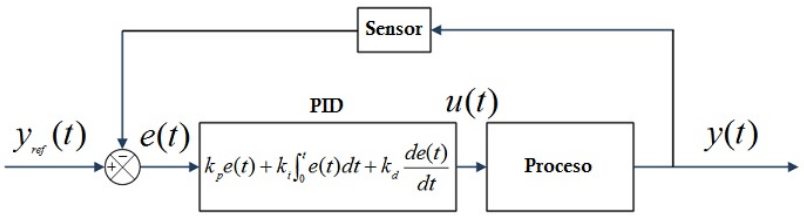

Figura 1.1: Esquema de control del PID.

Tomando como base la estructura de un PID clásico, y haciendo uso de la teoría Wavenet, en [**?**, **?**] se presenta un controlador PID Wavenet. Sin embargo, en esta tesis se utilizara una versión modificada del PID Wavenet presentada en [**?**], el PID Wavenet - Difuso.

1.1. Descripción de la problemática

El diseño de leyes de control depende del modelo dinámico de la planta a controlar y de su previa identificación. Básicamente existen dos formas de realizar la identificación de un sistema: en línea y fuera de línea. Cuando se realiza la identificación fuera de línea es necesario realizar exhaustivos experimentos y severas mediciones con equipo de laboratorio. En aplicaciones industriales no siempre es posible que los ingenieros de procesos o de planta cuenten con los equipos de medición o con el tiempo necesario para realizar los experimentos y mediciones necesarias que les permitan la identificación paramétrica del sistema. Por otro lado, el PID Wavenet que se trabaja en [**?**, **?**] requiere la sintonización de las tasas de aprendizaje las cuales se proponen de forma manual.

En el presente trabajo de tesis se plantea el siguiente problema: ¿Cómo explotar las características que ofrece la tecnología electrónica VLSI para algoritmos complejos de identificación de sistemas no lineales y control automático?

1.2. Descripción de la solución propuesta

La solución que se propone al problema planteado es, primeramente, seleccionar un algoritmo de control, que en este caso será el PID Wavenet difuso el cual se basa en el aprovechamiento que tienen las wavelets en conjunto con las RNA y la lógica difusa, para el control de sistemas no lineales. Para esto se emplea un algoritmo de identificación que hace uso de una red neuronal de base radial cuyas funciones de activación son wavelet hijas, donde el término no lineal de la identificación es empleado en la auto-sintonización de las ganancias del PID, además las tasas de aprendizaje de dichas ganancias serán inferidas por un modelo difuso.

Posteriormente, se probara el algoritmo de control PID Wavenet difuso en simulación, utilizando para este propósito el software Simulink para controlar un Motor de CD y el carro - péndulo invertido. A continuación, se codificara el algoritmo de control en lenguaje de descripción de hardware para su posterior implementación en un FPGA.

1.3. Objetivos de la tesis

Los objetivos del presente trabajo de tesis son los siguientes:

1.3.1. Objetivo general

Diseñar una arquitectura VLSI del controlador PID Wavenet Difuso.

1.3.2. Ojetivos específico

- Diseñar la arquitectura digital VLSI del PID Wavenet difuso mediante lenguaje de descripción de hardware VHDL.
- Validar en simulación el controlador PID Wavenet difuso para control del carro - péndulo invertido.
- Realizar análisis comparativo entre el PID clásico, PID Wavenet, PID Wavenet difuso.

1.4. Hipotesis

Haciendo uso de la lógica difusa y de la tecnología de los FPGAs es posible mejorar los resultados del PID Wavenet para el control de velocidad de un motor de CD y de posición angular del sistema carro - péndulo invertido.

1.5. Justificación

Los FPGAs son dispositivos digitales que ofrecen varias ventajas: son reconfigurables, permiten el procesamiento en paralelo, tienen bajo consumo de potencia, son de bajo costo para prototipos, entre otros. Debido a estas ventajas es idóneo implementar en un FPGA el diseño VLSI del controlador PID Wavenet Difuso. Esto debido a que en la industria, no siempre es posible para los ingenieros de procesos contar con los equipos o el tiempo necesario para realizar los experimentos y las severas mediciones requeridas para hacer la identificación paramétrica de un proceso a controlar.

1.6. Aportaciones

- Diseño de la arquitectura digital VLSI que integra una red neuronal de base radial cuyas funciones de activación son wavelets hijas, lógica difusa y un PID

discreto, que sea capaz de operar en tiempo real, con un consumo mínimo de energía y espacio.

- Implementación de la arquitectura en un FPGA.
- Publicación de los resultados obtenidos en congresos nacionales e internacionales y artículos sometidos a revistas indexadas.

1.7. Organizacion de la tesis

La presente tesis está organizada de la siguiente manera: En el Capítulo 2 Estudio del estado del arte, se discuten los trabajos previos que influyen en este trabajo de investigación. En el Capítulo 3 Controlador PID *Wavenet* Difuso, se describe de una forma detallada el funcionamiento del PID *Wavenet* Difuso, en el Capítulo 4 Resultados, se presentan los resultados obtenidos de las simulaciones al aplicar el controlador en un motor de CD y en el carro - péndulo invertido utilizando Simulink. Después de la validación del controlador, en el Capítulo 5 Diseño de la arquitectura VLSI, se muestran las características de la arquitectura VLSI creada. Posteriormente en el Capítulo 6 Resultados experimentales en FPGA, se presentan los resultados obtenidos al programar un FPGA con dicha arquitectura. Por ultimo en el Capítulo 7 Conclusiones, se discuten los resultados de este trabajo de investigación, y además, se enuncian algunos trabajos futuros relacionados a esta tesis.

Capítulo 2

Estado del Arte

En este capítulo se presenta el estado del arte sobre los trabajos de investigación que sirven como base de esta tesis. Están separados de acuerdo a su área de estudio correspondiente, las cuales, se pueden apreciar en la Figura **??**.

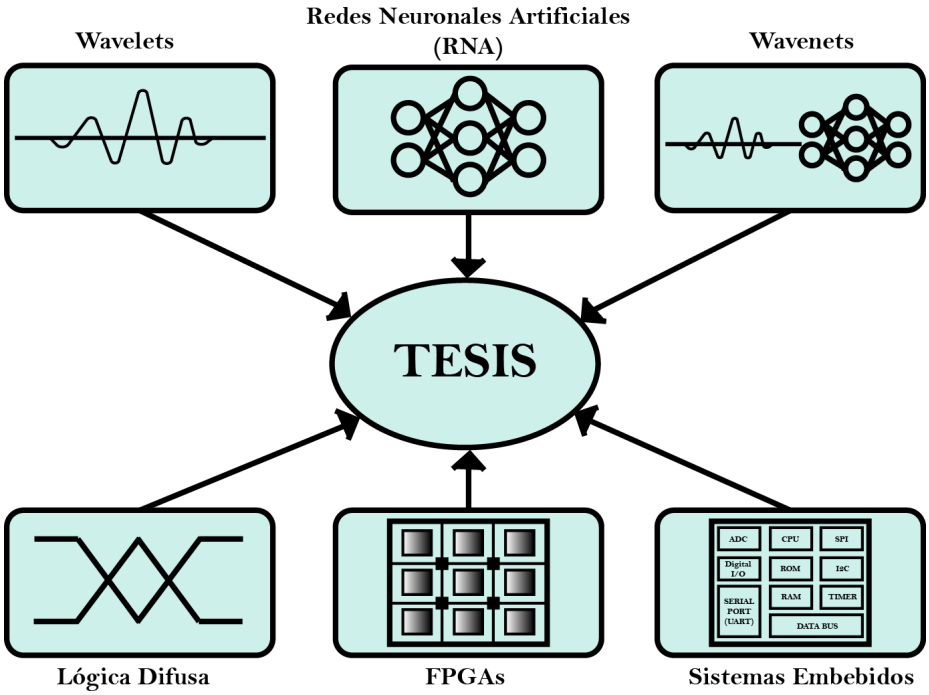

Figura 2.1: Áreas de investigacion requeridas para la tesis.

El capítulo está organizado de la siguiente manera: Primeramente, en la Sección 2.1 se da una pequeña introducción, mientras que en la Sección 2.2 se discute acerca de la teoría *wavelet*. En la Sección 2.3 se discuten los trabajos relevantes acerca de las RNA, esta área es esencial en esta tesis ya que sirve de enlace entre las demás áreas.

En la Sección 2.4 se discutirá acerca de la *wavenets*, implementadas en FPGAs. Por ultimo en la Sección 2.5 se abordan los trabajos relacionados a la implementación de controladores difusos en FPGAs.

2.1. Introducción

En este capítulo se presentan las principales aportaciones de las diversas áreas de estudio que están relacionadas de forma directa al desarrollo de este trabajo de tesis, las áreas de conocimiento involucradas en esta tesis se muestran en la Figura **??** y son las siguientes:

- Teoría *wavelet*.
- Redes Neuronales Artificiales.
- Teoría *wavenet*.
- Lógica difusa.
- Controladores PID.
- FPGAs.
- Sistemas Embebidos.

Una de las etapas fundamentales en el desarrollo de esta tesis es conocer los trabajos de investigación que de alguna forma interrelacionan las áreas de conocimiento propuestas en este trabajo de tesis y a partir de ellos, discernir como aprovechan las técnicas y herramientas de cada una de las áreas. Es importante mencionar que dentro de la literatura existe una gran cantidad de información relacionada a cada área, sin embargo, se trata de ser lo más concreto posible y así discutir los trabajos más relevantes sobre los cuales está basada esta tesis.

2.2. Teoría *wavelet*

En la actualidad existen una gran variedad de artículos, libros e incluso software que hacen de este tema toda una área de investigación. Por ejemplo los artículos de Daubechies [**?**, **?**, **?**] que son fuente de conocimiento en el área, así mismo están los artículos de Meyer [**?**, **?**, **?**]. En lo que respecta a libros se han publicado algunos con el objetivo de ser usados en función del área de aplicación, por ejemplo los libros [**?**, **?**] que están enfocados a dar las bases matemáticas sobre el tema, mientras que [**?**, **?**, **?**] presentan la teoría y sus diferentes aplicaciones en las áreas de ingeniería. En cuanto

a aplicaciones en control de sistemas dinámicos se tienen gran cantidad de publicaciones tanto en congresos internacionales como en revistas, como por ejemplo [**?**, **?**, **?**] y [**?**] donde se presentan aplicaciones de la teoría wavelet al procesamiento de señales médicas y reconocimiento de características faciales. Existen también diferentes trabajos que utilizan la transformada wavelet para: análisis multi resolución, aproximación de funciones, identificación de parámetros, sintonización PID, clasificación de perturbaciones, entre otras.

En [**?**] se presenta un controlador PID basado sobre análisis multi resolución basado en teoría wavelet. El controlador es similar a un controlador PID en cuanto a principio y aplicación. Dicho controlador es simulado para el control de velocidad de un servo sistema, el cual presenta resonancia de baja frecuencia, la cual es causada por los componentes de transmisión y el mal acoplamiento del motor con la carga, el objetivo es llevar al motor a una velocidad de referencia previamente dada. El controlador es probado de forma física en dicho sistema para controlar la posición, donde el objetivo es rotar la carga una revolución en un segundo sin sobre paso. De los resultados obtenidos en ambos casos, se resalta el hecho de que la señal de control es mucho más suave y sin cambios bruscos que utilizando un PID clásico. Aunque en ambos casos se lograron los objetivos cabe mencionar que se probó en sistemas lineales y con sus modelos matemáticos bien conocidos. En [**?**] hacen uso de la transformada Haar para un método computacional que determina las constantes de un control con retroalimentación del estado de un sistema con retardo y variante en el tiempo, mientras que en [**?**] utilizan un algoritmo de colocación sucesivo wavelet para resolver ecuaciones de Hamilton-Jacobi-Bellman generalizadas que aparecen en problemas de control. La transformada wavelet también es usada en un método adaptable en la simulación de circuitos no lineales [**?**], al igual que para desarrollar un algoritmo para estimación de la escala y el tiempo transcurrido entre dos señales recibidas y subsecuentemente para extraer el tiempo de retardo y la compresión doppler, específicamente en tiempo de retardo para movimiento lineal, en procesamiento de señales de radar y sonar [**?**].

2.3. Redes Neuronales Artificiales

Las redes neuronales artificiales (RNA) son unidades de procesamiento simples interconectadas masivamente inspiradas en los sistemas nerviosos biológicos, en especial el cerebro, sin embargo, no aproximan la complejidad de este órgano pero existen dos similitudes entre ambas. Primeramente, las unidades de procesamiento en ambas son simples unidades de cómputo y están interconectadas de forma masiva. Segundo, las conexiones entre las neuronas determinan el funcionamiento de la red. Las RNA presentan una estructura paralela y son implementadas en computadoras digitales convencionales, pero es ideal su implementación en dispositivos que cumplan con el

funcionamiento en paralelo, como son los circuitos integrados VLSI, los procesadores en paralelo y los dispositivos ópticos.

El interés de científicos e ingenieros durante muchos años por lograr maquinas inteligentes (que puedan emular el conocimiento humano), han llevado a desarrollar cada vez más las RNA, un estudio sistemático fue publicado por primera vez por McCulloch y Pitts [**?**] donde se presenta el primer modelo matemático de una neurona, mostrando que una simple red neuronal puede ser calculada por funciones aritméticas o lógicas, cuatro años después los mismos autores exploraron los paradigmas de la red para un reconocimiento de patrones usando un perceptrón de una simple capa [**?**]. La primera aplicación de las RNA se da a finales de los 50's con la invención del perceptrón y una regla de aprendizaje asociativa, expuestas por Frank Rosenbaltt [**?**]. Casi al mismo tiempo Bernard Widrow y Ted Hoff [**?**] muestran otro algoritmo de aprendizaje que se utilizó para entrenar una red neuronal lineal adaptable que era de estructura y capacidad parecida al perceptrón de Rosenbaltt, dicha ley de aprendizaje aun es utilizada hoy en día. En 1972 Teuvo Kohonen [**?**] y James Anderson [**?**] desarrollaron independientemente una nueva red neuronal que podía actuar como memoria. Durante la década de los 80's el estudio de las RNA incrementó en forma considerable por consecuencia de la aparición de nuevas computadoras personales, entonces, dos nuevos conceptos fueron introducidos a este campo. Primero, utilizar mecanismos estadísticos para explicar la operación de las RNA recurrentes, que podían ser usadas como memorias asociativas, las cuales, fueron descritas por John Hopfield [**?**]. El segundo concepto fue el algoritmo de retropropagación para entrenar redes de perceptrón multicapa, el cual fue descubierto de manera independiente por diferentes investigadores, entre los trabajos más influyentes de este algoritmo de aprendizaje son los de David Rumelhart y James McClelland [**?**]. Desde ese entonces y hasta la actualidad se han publicado libros y artículos relacionados con este tema, por mencionar algunos, están los libros [**?**, **?**, **?**, **?**].

2.4. *Wavenets*

Una red neuronal posee: una capa de entrada, una o varias capas ocultas y una capa de salida, cada capa posee varias neuronas las cuales tienen una función de activación: función sigmoidal o función tangente hiperbólica, que son las más comúnmente empleadas [**?**]. Pero existen trabajos de investigación donde dicha función de activación es sustituida por una función wavelet, por lo que es conocida como wavenet.

Existen muchos trabajos reportados donde hacen uso de las wavenets para diferentes propósitos, por ejemplo, en [**?**] utilizan funciones wavelets Gaussianas como funciones de activación para diseñar un controlador adaptable para robots. Otra aplicación de las wavenets es presentada en [**?**], donde un control PID es auto-sintonizado

empleando una wavenet. Además, pueden utilizarse para identificación de parámetros de sistemas dinámicos no lineales, como en [**?**], donde se utilizan dos estructuras wavenet: una para identificación y otra para sintonizar un PID. Debido a que las Wavenet combinan las ventajas ofrecidas por el aprendizaje de una red neuronal y la representación de una wavelet, ofrecen una aproximación eficiente en sistemas de control dinámicos que usualmente poseen complejidad, como no linealidades e incertidumbres.

Otro uso de las wavenets es en control inteligente, para obtener más robustez en un sistema donde se utiliza un vector de control en máquinas de inducción [**?**], y para el aprendizaje en línea y la cancelación de errores repetitivos en lectura de unidad de disco [**?**].

En [**?**, **?**, **?**, **?**] hacen uso de wavenets para el control adaptable en sistemas dinámicos no lineales. Los resultados obtenidos basados en simulaciones en modelos perturbados como el péndulo invertido y otros modelos de sistemas no lineales, demuestran la eficacia de este tipo de control adaptable.

Existen algunas tesis previas a este trabajo de investigación, donde se emplean las wavenets para la aproximación de funciones desconocidas, obteniéndose buenos resultados experimentales y en simulación numérica [**?**, **?**]. En [**?**] se presenta un esquema de aprendizaje reforzado derivado del método Q-Learning, el cual emplea RNAs, cuyas funciones de activación son funciones wavelets y los algoritmos desarrollados son válidos para la aplicación de control de un sistema subactuado.

2.5. Controladores Difusos

Los controladores difusos (Fuzzy Logic Controller FLC) ofrecen un método simple para controlar sistemas imprecisos y altamente complejos, esto sin la necesidad de tener un modelo matemático del sistema a controlar. La lógica difusa nos provee una alternativa para lidiar con las incertidumbres de un sistema usando un método que se asemeja al razonamiento humano y mediante el uso de expresiones lingüísticas que son evaluadas por un conjunto de reglas del tipo IF - THEN(un proceso similar al razonamiento humano [**?**, **?**]). Esta similitud, permite al controlador capturar el conocimiento obtenido por la experiencia humana y transformarlo en una serie de reglas para controlar el proceso.

Debido a las ventajas que ofrecen los controladores difusos, el desarrollo e investigación de estos sistemas ha crecido en una manera impresionante y para esto se ha utilizado diversas técnicas de hardware y software [**?**, **?**]. Estas técnicas se clasifican generalmente en dos categorías:

- Soluciones de software, utilizando microprocesadores de propósito general y ASIPs (Application Specific Instruction Processors.

- Soluciones de Hardware, haciendo uso de procesadores difusos y ASICs (Application Specific Integrated Circuits)

Un trabajo muy interesante en esta área es presentado en [**?**], en este trabajo se presenta un controlador difuso adaptable implementado en un FPGA, la relevancia de este trabajo recae en el hecho de que sus funciones de membresía y base de reglas difusas pueden ser modificadas en tiempo real, este trabajo fue implementado en un FPGA de la familia Virtex-II. Otro trabajo sobresaliente es presentado en [**?**] donde proponen una técnica de división del conjunto de reglas para reducir el consumo de potencia del controlador difuso, a su vez desarrollaron e implementaron de manera exitosa la metodología para reducir el consumo de potencia del controlador. La implementación fue realizada en un FPGA de la familia Kintex 7 y se usó el lenguaje Verilog-HDL con el software ISE 14.2 de Xilinx.

2.6. FPGAs

Desde su creación en 1984, la implementación de los FPGAs ha ido en aumento, hoy en día conforman un elemento vital para el desarrollo de prototipos así como para la investigación. Cada día son más las aplicaciones y las áreas donde se ven involucrados estos dispositivos ejemplos de estas áreas son: control, DSPs, reconocimiento de voz, reconocimiento de imágenes, instrumentación biomédica, aplicaciones de aeronáutica y defensa, entre otras.

Un FPGA (Field Programmable Gate Array) es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad pueden ser reconfiguradas mediante un lenguaje de descripción especializado. Los FPGAs son capaces de reproducir funciones tan simples como las compuertas lógicas o sistemas combinacionales hasta sistemas tan complejos como los SoC (System on Chip).

Los FPGAs tienen las ventajas de que son reprogramables, lo cual implica una gran flexibilidad en el flujo de diseño, sus bajos costos de desarrollo y adquisición, y que el tiempo de desarrollo es mucho menor a otros dispositivos como los ASICs.

2.7. Sistemas Embebidos

Hoy en día existe un gran avance en el campo de los sistemas digitales, debido a este, la complejidad de dichos sistemas ha crecido en gran manera. Día con día son más las aplicaciones que involucran el uso de sistemas digitales, en algunas de ellas, se requiere de una gran tolerancia a fallas y desempeños muy altos, ya que un error en estas aplicaciones, se resume en costos muy elevados y en algunas ocasiones, incluso, vidas humanas.

Aun en la actualidad, en algunos países en vía de desarrollo, se utilizan los procesadores de 8 o 12 bits como plataforma de desarrollo para un amplio número de aplicaciones, sin embargo, debido a las limitaciones de esta plataforma tales como velocidad, periféricos, número de pines, ancho de bus de datos, librerías, etc., no es posible realizar aplicaciones complejas que requieran procesamiento en tiempo real o multitarea, además, el tiempo de desarrollo en estas plataformas depende fuertemente de la experiencia del diseñador.

Gracias al avance continuo en los sistemas de software y componentes de hardware, hoy en día existen un sinfín de nuevas oportunidades para la implementación de los sistemas embebidos en diversos campos tales como: control, aplicaciones de red, instrumentación biomédica, aplicaciones de seguridad, aplicaciones militares, etc. En la actualidad existen una gran variedad de dispositivos de hardware de bajo costo (FPGAs, CPLDs, IPs e ICs) y un sinfín de librerías que facilitan el control de dispositivos de hardware como LCDs, interfaces USB, RS232, librerías matemáticas, científicas, etc. que permiten generar programas y aplicaciones sumamente complejos.

Un sistema embebido es un sistema desarrollado para controlar una función o un rango de funciones, este sistema emplea una combinación de hardware y software, normalmente este sistema es parte de un sistema más grande que podría no ser una computadora y que trabaja en un ambiente reactivo con restricciones temporales. A diferencia de los procesadores de propósito general que están diseñados para cubrir un amplio rango de necesidades, los sistemas embebidos se diseñan para cubrir necesidades específicas. En este tipo de sistemas el software tiene como objetivo proporcionar flexibilidad y funcionalidad. El hardware, por otro lado, es utilizado para proporcionar desempeño y a veces seguridad.

Capítulo 3

Controlador PID *wavent*-difuso

El PID *wavenet*-difuso es un algoritmo de control que sintoniza en línea las ganancias proporcional, integral y derivativa de un PID discreto, esto se logra mediante la identificación de un sistema dinámico no lineal a controlar donde, se asume, el modelo matemático es desconocido. Dicha identificación es posible gracias a una red neuronal de base radial cuyas funciones de activación son *wavelets* hijas y un filtro de respuesta infinita al impulso (IIR) en cascada.

En este capítulo se presenta el controlador PID *wavenet*. El presente capítulo se presenta de la siguiente manera: en la Sección 3.1 se presenta una breve introducción al controlador PID *wavenet* partiendo desde su estructura básica de funcionamiento, en la Seccion 3.2 se presenta el algoritmo de control de dicho controlador, esta sección es dividida en tres etapas: 3.2.1 Identificación del sistema; 3.2.2 Controlador PID discreto y 3.2.3 Auto-sintonización de las ganancias del PID. En la Sección 3.3 se muestra el algoritmo de control que es programado en *Simulink* y finalmente en la Sección 3.4 se hacen algunos comentarios sobre este capítulo.

3.1. Introducción

Desde su creación en las primeas decadas del siglo XX hasta la actualidad, el controlador PID ha sido uno de los controladores más usados dentro de la industria, esto se debe a su gran facilidad de manejo y su gran desempeño dentro de las areas de control industrial. Sin embargo, el algoritmo del controlador PID clásico dificilmente logra contener el seguimiento de referencia cuando el proceso que se pretende controlar presenta dinámicas complejas como zonas muertas o caracteristicas no lineales. El PID basa su funcionamiento en actuar de manera proporcional, integral y derivativa sobre una señal de error $e(t)$ definida como la diferencia existente entre la señal de referencia $y_{ref}(t)$ y la señal de salida del proceso $y(t)$. De esta manera se logra generar una ley de control $u(t)$ que manipula la salida del proceso en una forma deseada.

Existen diversas técnicas analíticas y experimentales para sintonizar las ganancias del PID, sin embargo, es muy dificil lograr sintonizar estas ganancias cuando la planta a controlar presenta características altamente no lineales, una alternativa a este problema de sintonización es el algoritmo de control denominado PID *wavenet*, este algoritmo de control estima la salida del sistema a controlar mediante una *wavenet* y un filtro IIR, esta estimación es utilizada para sintonizar las ganancias de un PID discreto, todo lo anteriormente mencionado se realiza en linea[].

3.2. PID *wavenet*

El esquema de control del PID *wavenet* que se implementa en este trabajo de tesis se muestra en la Figura **??**. Se puede identificar que la arquitectura de este controlador consta de tres etapas principales para poder manipular la salida del sistemas en forma deseada: Identificación del sistema dinámico, Controlador PID clásico y Auto-sintonización de ganancias del PID.

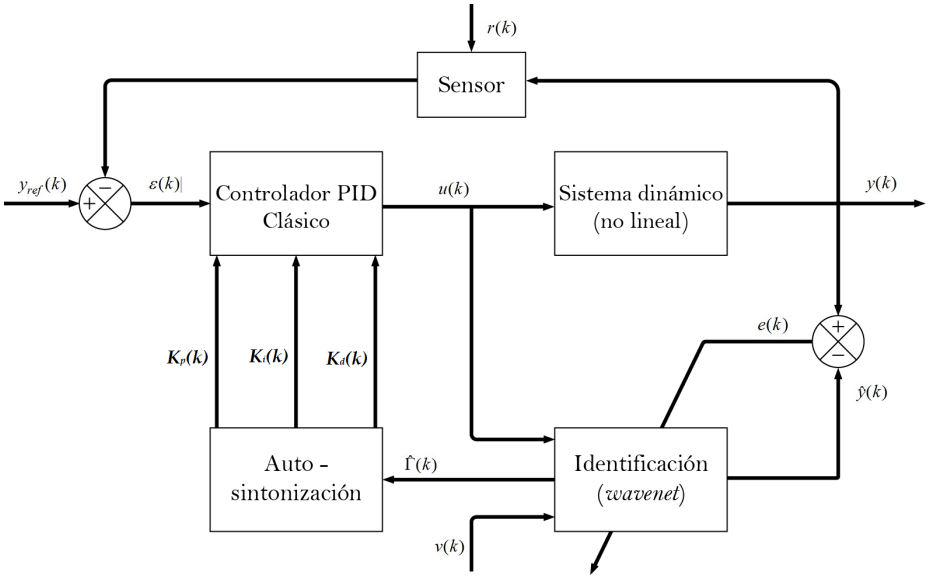

Figura 3.1: Diagrama a bloques del controlador PID *wavenet*.

3.2.1. Identificación del sistema

El proceso de identificación se logra a travez de una red neuronal de base radial en la que sus funciones de activación $\psi(\tau)$ son funciones *wavelet* hijas $\psi_l(\tau_l)$. Para lograr reducir el número de iteraciones en el proceso de aprendizaje, se cuenta también con un filtro IIR, el cual tiene como proposito filtrar las neuronas que tienen muy poca

contribución al proceso de identificación[]. Estos elementos antes mencionados son ilustrados en las Figuras **??**.

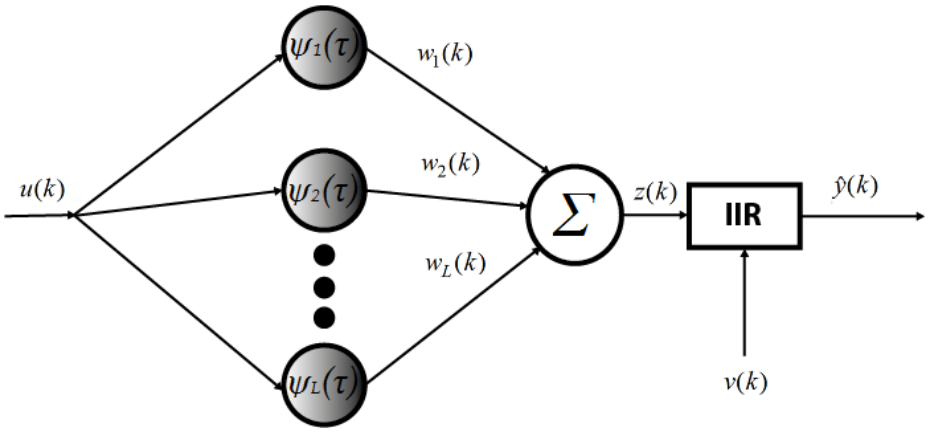

Figura 3.2: Diagrama de la red neuronal *wavenet* con filtro IIR en cascada.

donde $\psi_l(\tau) = \psi_l \frac{k-b_l}{a_l}, 1 \leq l \leq L$, es una *wavelet* hija.

La función *wavelet* $\psi(\tau)$ es conocida como *wavelet* madre, porque diferentes funciones son generadas a partir de ella, por su dilatación ó contracción y translación, llamadas *wavelets* hijas $\psi_{a,b}(\tau)$, representadas matemáticamente como[]:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\tau) \tag{3.1}$$

con $a \neq 0$; $a, b \in \mathbb{R}$ y

$$\tau = \frac{k-b}{a} \tag{3.2}$$

donde a es la variable de escala, la cual permite dilataciones y contracciones, b es la variable de translación, que permite desplazamientos en el instante k. La representación matemática y la derivada parcial con respecto de b de algunas *wavelets* madre son mostradas en las tablas **??**, respectivamente

wavelet madre	$\psi(\tau)$
Morlet	$\cos(\omega_0\tau)e^{-0.5\tau^2}$
RASP1	$\frac{\tau}{(\tau^2+1)^2}$
RASP2	$\frac{\tau\cos(\tau)}{\tau^2+1}$
RASP3	$\frac{\sin(\pi\tau)}{\tau^2+1}$
POLYWOG1	$\tau e^{\frac{\tau^2}{2}}$
POLYWOG2	$(\tau^3-3\tau)e^{-\frac{\tau^2}{2}}$
POLYWOG3	$(\tau^4-6\tau^2+3)e^{-\frac{\tau^2}{2}}$
POLYWOG4	$(1-\tau^2)e^{-\frac{\tau^2}{2}}$
POLYWOG5	$(3\tau^2-\tau^4)e^{-\frac{\tau^2}{2}}$
Shannon	$\frac{\sin(2\pi\tau)-\sin(\pi\tau)}{\pi\tau}$

Tabla 3.1: Representación matemática de algunas *wavelets* madre.

La señal de aproximación de la *wavenet* con filtro IIR $\hat{y}(k)$ puede ser calculada como:

$$\hat{y}(k)=\sum_{i=0}^{M}c_i z(k-i)u(k)+\sum_{j=1}^{N}d_j\hat{y}(k-j)v(k) \tag{3.3}$$

donde

$$z(k)=\sum_{l=1}^{L}\omega_l\psi_l(k) \tag{3.4}$$

L es el número de *wavelets* hijas, ω_l son los pesos de cada neurona en la *wavenet*, c_i y d_j son los coeficientes de adelanto y atraso del filtro IIR, respectivamente. M y N representan el número de los coeficientes de adelanto y atraso de dicho filtro, respectivamente. Los parámetros de la *wavenet* en forma matricial son:

$$\mathbf{A}(k)\triangleq[a_1(k),a_2(k),\cdots,a_L(k)]^T \tag{3.5}$$
$$\mathbf{B}(k)\triangleq[b_1(k),b_2(k),\cdots,b_L(k)]^T \tag{3.6}$$
$$\mathbf{W}(k)\triangleq[w_1(k),w_2(k),\cdots,w_L(k)]^T \tag{3.7}$$

y los parámetros del filtro IIR, representados de igual forma:

$$\mathbf{C}(k)\triangleq[c_1(k),c_2(k),\cdots,c_M(k)]^T \tag{3.8}$$
$$\mathbf{D}(k)\triangleq[d_1(k),d_2(k),\cdots,b_N(k)]^T \tag{3.9}$$

todos estos parámetros son optimizados por medio de un algoritmo de aprendizaje basado en mínimos cuadrados medios (LMS), tras minimizar una función de costo $\mathbf{E}$,

definida como

$$\mathbf{E} = \frac{1}{2}\sum_{k=1}^{T} e^2(k) \tag{3.10}$$

donde el error de estimación $e(k)$ entre la salida de la *wavenet* con filtro IIR $\hat{y}(k)$ y la salida real del sistema $y(k)$, es

$$e(k) = y(k) - \hat{y}(k) \tag{3.11}$$

Para minimizar $\mathbf{E}$ se aplica el método del gradiente de pasos descendentes, que utiliza las siguientes derivadas

$$\frac{\partial \mathbf{E}}{\partial w_l(k)} = -e(k)\mathbf{C}(k)^T \Psi_l(\tau_l) u(k) \tag{3.12}$$

$$\frac{\partial \mathbf{E}}{\partial b_l(k)} = -e(k)\mathbf{C}(k)^T \Psi_{b_l}(\tau_l) w_l(k) u(k) \tag{3.13}$$

$$\frac{\partial \mathbf{E}}{\partial a_l(k)} = \tau_l \frac{\partial \mathbf{E}}{\partial b_l(k)} \tag{3.14}$$

$$\frac{\partial \mathbf{E}}{\partial c_m(k)} = -e(k) z(k-M) u(k) \tag{3.15}$$

$$\frac{\partial \mathbf{E}}{\partial d_n(k)} = -e(k) \hat{y}(k-N) v(k) \tag{3.16}$$

donde

$$\Psi_l(\tau) = [\psi_l(\tau),\, \psi_l(\tau-1), \cdots,\, \psi_l(\tau-M)]^T \tag{3.17}$$

$$\Psi_{b_l} = \left[\frac{\partial \psi_l(\tau)}{\partial b_l(k)},\, \frac{\partial \psi_l(\tau-1)}{\partial b_l(k)}, \cdots,\, \frac{\partial \psi_l(\tau-M)}{\partial b_l(k)}\right] \tag{3.18}$$

La actualización de los parámetros cumple con la siguiente regla

$$\Delta\theta(k) = -\frac{\partial \mathbf{E}}{\partial \theta(k)} \tag{3.19}$$

$$\theta(k+1) = \theta(k) + \mu_\theta \Delta\theta(k) \tag{3.20}$$

donde θ puede ser cualquiera de los parámetros ajustados: $\mathbf{W}(k)$, $\mathbf{A}(k)$, $\mathbf{B}(k)$, $\mathbf{C}(k)$ o $\mathbf{D}(k)$. La variable $\mu_\theta \in \mathbb{R}$ representa el coeficiente de aprendizaje para cada uno de los parámetros ajustados.

3.2.2. Identificación y control

Considerando un sistema dinámico no lineal, el cual es descrito por las siguientes ecuaciones de estado discretas[]

$$x(k+1) = f\,[x(k), u(k), k] \tag{3.21}$$

$$y(k) = g\,[x(k), k] \tag{3.22}$$

donde $x \in \mathbb{R}^n$, $u, y \in \mathbb{R}$ y

$$f, g \in \mathbb{C} \tag{3.23}$$

son funciones no lineales que, se asume, son desconocidas. La entrada $u(k)$ y la salida del sistema $y(k)$ son los únicos datos disponibles. Si el sistema linealizado alrededor del punto de equilibrio es observable, existe una representación entrada - salida del mismo dada por[]

$$y(k+1) = \beta[\mathbf{Y}(k), \mathbf{U}(k)] \tag{3.24}$$

donde

$$\mathbf{Y}(k) = [y(k), y(k-1), \cdots, y(k-n+1)] \tag{3.25}$$

$$\mathbf{U}(k) = [u(k), u(k-1), \cdots, u(k-n+1)] \tag{3.26}$$

Dicho de otra manera, existe una función β que mapea la salida $y(k)$, la entrada $u(k)$ y sus $n-1$ valores pasados en $y(k+1)$. Un modelo alternativo de una planta desconocida que puede simplificar el algoritmo de control es el siguiente:

$$y(k+1) = \Phi[\mathbf{Y}(k), \mathbf{U}(k)] + \Gamma[\mathbf{Y}(k), \mathbf{U}(k)] \cdot u(k) \tag{3.27}$$

si los términos Φ y Γ son exactamente conocidos, entonces la señal de control $u(k)$ para obtener la salida deseada $y_{ref}(k+1)$ es:

$$u(k) = \frac{y_{ref}(k+1) - \Phi[\mathbf{Y}(k), \mathbf{U}(k)]}{\Gamma[\mathbf{Y}(k), \mathbf{U}(k)]} \tag{3.28}$$

Sin embargo, los términos Φ y Γ son desconocidos. Por lo tanto, se utiliza una red neuronal *wavenet* para poder aproximar las dinámicas del sistema de la siguiente manera:

$$\hat{y}(k+1) = \hat{\Phi}[y(k), \Theta_\Phi] + \hat{\Gamma}[y(k), \Theta_\Gamma] \cdot u(k) \tag{3.29}$$

ahora bien, comparando la expresion anterior (3.29) con la salida estimada de la wavenet con el filtro IIR (3.3) se obtiene que:

$$\hat{\Phi}[y(k), \Theta_\Phi] = \sum_{j=1}^{N} d_j \hat{y}(h-j) v(k) \tag{3.30}$$

$$\hat{\Gamma}[y(k), \Theta_\Gamma] = \sum_{i=0}^{M} c_i z(k-i) \tag{3.31}$$

$$z(k) = \sum_{l=1}^{L} w_l \psi_l(k) \tag{3.32}$$

Por lo tanto, si ambas no linealidades Φ y Γ son estimadas por las dos funciones de la *wavenet* $\hat{\Phi}$ y $\hat{\Gamma}$ con sus respectivos parámetros ajustables conocidos como Θ_Φ y Θ_Γ, la señal de control del PID que sigue a la referencia deseada $y_{ref}(k)$ puede ser calculada como[]:

$$u(k+1) = u(k)+k_p(k)[\varepsilon(k)-\varepsilon(k-1)]+k_i(k)\varepsilon(k)+k_d[\varepsilon(k)-2\varepsilon(k-1)+\varepsilon(k-2)] \quad (3.33)$$

donde $k_p(k)$, $k_i(k)$ y $k_d(k)$ son las ganacias proporcional, integral y derivativa del controlador PID, $u(k)$ es la entrada de control a la planta en el instante k, y $\varepsilon(k)$ es el error de seguimiento dado por:

$$\varepsilon(k) = y_{ref}(k) - y(k) \quad (3.34)$$

3.2.3. Auto-sintonización de las ganancias del controlador PID

Las ganancias de ponderación del controlador PID discreto se determinan por medio de un modelo basado en reglas difusas. Para esto se considera el diagrama a bloques dado en la Figura, mostrándose dos niveles de control, en uno de los bloques se encuentra el controlador PID convencional y en el otro bloque correspondiente al segundo nivel representa el sistema difuso que ajusta las ganancias de ponderación del PID.

Como las ganancias del controlador $k_p(k)$, $k_i(k)$ y $k_d(k)$ han sido consideradas en la función de costo $\mathbf{E}$ (3.10) estas pueden ser actualizadas de la siguiente forma:

$$k_p(k) = k_p(k-1) + \mu_p e(k)\hat{\Gamma}(k)[\varepsilon(k) - \varepsilon(k-1)] \quad (3.35)$$

$$k_i(k) = k_i(k-1) + \mu_i e(k)\hat{\Gamma}(k)\varepsilon(k) \quad (3.36)$$

$$k_d(k) = k_d(k-1) + \mu_d e(k)\hat{\Gamma}(k)[\varepsilon(k) - 2\varepsilon(k-1) + \varepsilon(k-2)] \quad (3.37)$$

donde $\Gamma(k)$ es la parte de la identificación del sistema descrita por (3.31) y los parámetros μ_p, μ_i y μ_d son las ganancias de ponderación del controlador PID. Los rangos de las ganancias de ponderación se pueden determinar como: $[\mu_{p_{\text{mín}}} : \mu_{p_{\text{máx}}}]$, $[\mu_{i_{\text{mín}}} : \mu_{i_{\text{máx}}}]$ y $[\mu_{d_{\text{mín}}} : \mu_{d_{\text{máx}}}]$, estos rangos som determinados a prueba y error, realizando para este fin una serie de simulaciones numéricas varindo dichas ganancias y observando el comportamiento del sistema en lazo cerrado.

Las ganancias de ponderacion μ_p, μ_i y μ_d satisfacen respectivamente que:

$$\mu_p \in [\mu_{p_{\text{mín}}} : \mu_{p_{\text{máx}}}] \quad (3.38)$$

$$\mu_i \in [\mu_{i_{\text{mín}}} : \mu_{i_{\text{máx}}}] \quad (3.39)$$

$$\mu_d \in [\mu_{d_{\text{mín}}} : \mu_{d_{\text{máx}}}] \quad (3.40)$$

Por conveniencia μ_p, μ_i y μ_d son normalizadas para que tengan un rango de cero a uno con las siguientes transformaciones lineales [**?**]:

$$\mu'_p = \frac{\mu_p - \mu_{p_{\text{mín}}}}{\mu_{p_{\text{máx}}} - \mu_{p_{\text{mín}}}} \tag{3.41}$$

$$\mu'_i = \frac{\mu_i - \mu_{i_{\text{mín}}}}{\mu_{i_{\text{máx}}} - \mu_{i_{\text{mín}}}} \tag{3.42}$$

$$\mu'_d = \frac{\mu_d - \mu_{d_{\text{mín}}}}{\mu_{d_{\text{máx}}} - \mu_{d_{\text{mín}}}} \tag{3.43}$$

donde μ'_p, μ'_i y μ'_d son los parámetros sintonizados por el sistema difuso.

El sistema difuso consiste en un módulo de dos entradas y tres salidas como se muestra en la Figura **??**.

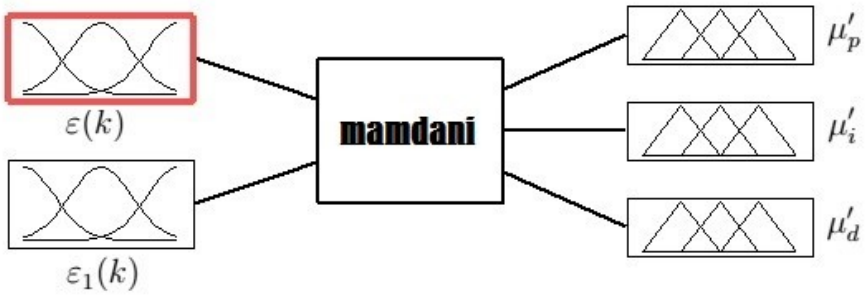

Figura 3.3: Sistema difuso.

donde $\varepsilon(k)$ es el error de seguimiento y $\varepsilon_1(k)$ es la aproximación de su derivada y se calcula como:

$$\varepsilon_1(k) = \varepsilon(k) - \varepsilon(k-1) \tag{3.44}$$

Las reglas difusas tienen la estructura del tipo SI-ENTONCES, que se muestra a continuación:

$$\mathcal{R}_i : \textbf{SI}\ \varepsilon(k)\ \textbf{es}\ A^i\ \textbf{y}\ \varepsilon_1(k)\ \textbf{es}\ B^i \quad \textbf{ENTONCES} \quad \mu'_p = C^i,\ \mu'_i = D^i,\ \mu'_d = E^i \tag{3.45}$$

donde A^i, B^i, C^i, D^i y E^i son los conjuntos difusos de la i-ésima regla, con $i = 1, 2, \cdots, M$.

Los conjuntos difusos y las funciones de membresía para cada variable se definen como se muestra en la Figura **??**

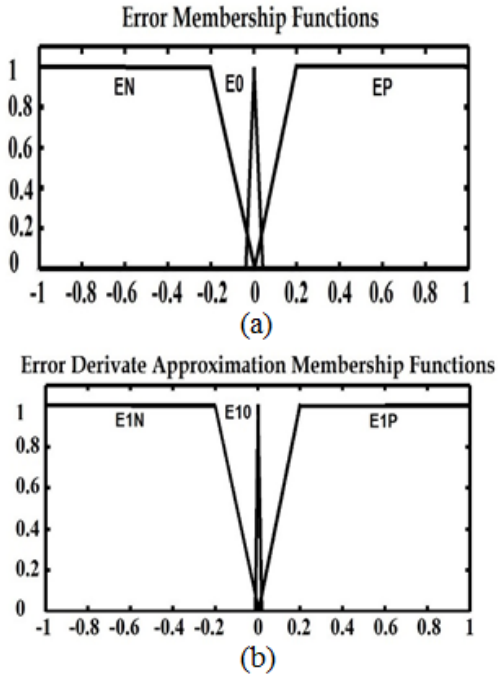

Figura 3.4: Función de membresía del error (a) y su derivada (b)

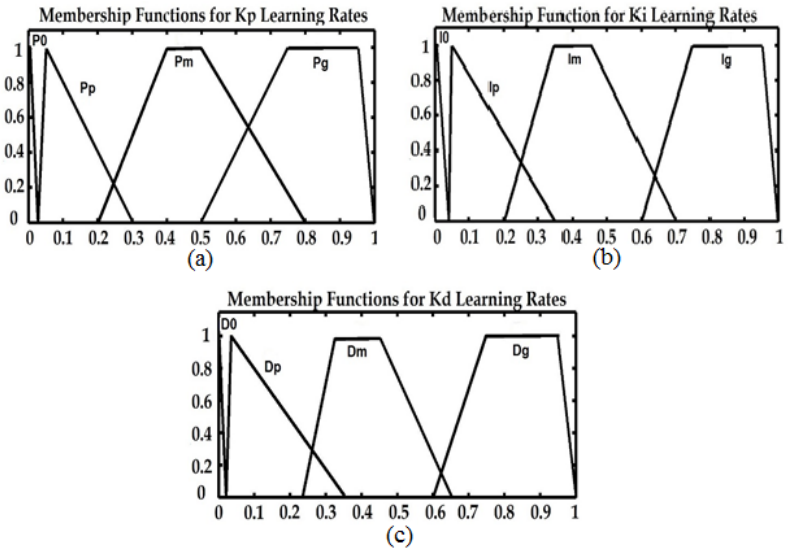

Figura 3.5: Función de membresía de las variables μ_p (a) μ_i (b) y μ_d (c)

El sistema difuso resultante esta formado por 9 reglas, las cuales son mostradas en la Tabla **??**.

ε	ε_1	μ_p	μ_i	μ_d
EN	E1N	Pg	Ig	Dg
EN	E1Z	Pm	Im	Dm
EN	E1P	Pp	Ip	Dp
EZ	E1N	Pp	Ip	Dp
EZ	E1Z	Pz	Iz	Dz
EZ	E1P	Pp	Ip	Dp
EP	E1N	Pp	Ip	Dp
EP	E1Z	Pm	Im	Dm
EP	E1P	Pg	Ig	Dg

Tabla 3.2: Reglas difusas para ajuste de ganancias de ponderacion del PID.

donde 'N', 'Z/z', 'P', 'p', 'm' y 'g' representan *Negativo, Cero, Positivo, Pequeño, Mediano y Grande*, respectivamente.

3.3. Algoritmo

En esta sección se describe el algoritmo del contolador PID *wavenet*-difuso empleado en las simulaciones numéricas, en la plataforma de *Simulink* y dentro de la configuración del FPGA.

	ALGORITMO del controlador PID *wavenet*-difuso.
1.	Establecer las condiciones iniciales de $\mathbf{W}(k)$, $\mathbf{A}(k)$, $\mathbf{B}(k)$, $\mathbf{C}(k)$ y $\mathbf{D}(k)$
2.	Inicializar con cero los valores anteriores que dependen N y M del filtro IIR (para el caso de $M = 2$ y $N = 2$) $z(k-2)$, $z(k-1)$, $\hat{y}(k-2)$, $\hat{y}(k-1)$, $\Psi_{b_l}(\tau-2)$, $\Psi_{b_l}(\tau-1)$, $\Psi_b(\tau-2)$, $\Psi_b(\tau-2)$.
3.	Inicializar con cero $\varepsilon(k-2)$, $\varepsilon(k-1)$. Dar condiciones iniciales a $k_p(k-1)$, $k_i(k-1)$ y $k_d(k-1)$
4.	Fijar las tazas de aprendizaje μ_W, μ_A, μ_B, μ_C, μ_D y el valor de la señal de persistencia del filtro IIR $v(k)$. Dar condiciones iniciales a μ_p, μ_i, μ_d.
5.	Adquirir los valores de la entrada $u(k)$ y la salida $y(k)$ de la planta. Fijar la referencia deseada $y_{ref}(k)$. Inicializar con cero el contador de las épocas del algoritmo de aprendizaje.
6.	Calcular las *wavelets* hijas y su derivada respecto a $b(k)$, Tablas **??**.
7.	Calcular la salida de la *wavenet* $z(k)$, ecuación (3.4).
8.	Calcular la salida de la *wavenet* con filtro IIR $\hat{y}(k)$, ecuación (3.3).
9.	Calcular el error de estimación $e(k)$, ecuación (3.11).
10.	Calcular los gradientes, ecuaciones (3.12), (3.13), (3.14), (3.15) y (3.16).
11.	Calcular incrementos, ecuaciones (3.19) y (3.20).
12.	Realizar los corrimientos respectivos a $\mathbf{W}(k) = \mathbf{W}(k+1)$, $\mathbf{A}(k) = \mathbf{A}(k+1)$, $\mathbf{B}(k) = \mathbf{B}(k+1)$, $\mathbf{C}(k) = \mathbf{C}(k+1)$, $\mathbf{D}(k) = \mathbf{D}(k+1)$, $z(k-2) = z(k-1)$, $z(k-1) = z(k)$, $\hat{y}(k-2) = \hat{y}(k-1)$, $\hat{y}(k-1) = \hat{y}(k)$, $\Psi_{b_l}(\tau-2) = \Psi_{b_l}(\tau-1)$, $\Psi_{b_l}(\tau-1) = \Psi_{b_l}(\tau)$, $\Psi_b(\tau-2) = \Psi_b(\tau-1)$, $\Psi_b(\tau-1) = \Psi_b(\tau)$.
13.	Incrementar las épocas. Si aún no se han completado el número de épocas del proceso de aprendizaje regresar al paso 6. De lo contrario ir al paso siguiente.
14	Calcular $\hat{\Gamma}$, ecucación (3.32). Calcular el error de seguimiento $\varepsilon(k)$ y su aproximacion de la derivada ε_1, ecuaciones (3.34) y (3.44).
15.	Obtener las ganancias de ponderación μ_p, μ_i y μ_d del PID.
16.	Sintonizar las ganancias del PID discreto, ecuaciones (3.35), (3.36) y (3.37).
17.	Calcular la señal de control del PID $u(k)$, ecuación (3.34).
18.	Realizar corrimientos respectivos a $\varepsilon(k-2) = \varepsilon(k-1)$, $\varepsilon(k-1) = \varepsilon(k)$, $k_p(k-1) = k_p(k)$, $k_i(k-1) = k_(k)$, $k_d(k-1) = k_d(k)$.
19.	Regresar al paso 5 e incrementar la iteración k.

Tabla 3.3: Algoritmo del controlador PID *wavenet*-difuso.

Capítulo 4

Desarrollo del sistema embebido

La arquitectura de hardware del sistema embebido diseñada para desempeña el funcionamiento del PID *wavenet*-difuso, es presentada en este capítulo. Se trata de hacer énfasis en el paralelismo de dicha arquitectura, ya que esta es basada en una RNA, también se resalta el hecho de que no existe dependencia alguna de tarjetas de adquisición de datos externas o softwares adicionales para dicha arquitectura. La plataforma que ha sido seleccionada para lograr este objetivo es el FPGA *Spartan 3E Starter kit* de Xilinx.

El presente capítulo se organiza de la siguiente manera: Inicialmente, en la Sección 4.1 se da una breve introducción de la arquitectura de hardware diseñada. Posteriormente en la Sección 4.2 se describe la parte de la arquitectura que corresponde a la adquisición de la señal análoga proveniente del sensor. Después, en la Sección 4.3, se explica la estructura que corresponde al controlador PID *wavenet*. Por ultimo en la Sección 4.4 se explicara la sección de la DAC encargada de convertir nuestra señal de control digital a una señal análoga.

4.1. Intoducción

En forma general, el funcionamiento del controlador PID *wavenet*-difuso dentro de un FPGA, se puede describir como una gran máquina de estados finitos (FSM) compuesta por cinco etapas principales, donde cada etapa en si es una FSM más pequeña, se adopta este esquema de máquinas de estados debido a que facilita en gran manera el cumplir con los requerimientos temporales del sistema, estos requerimientos temporales se presentan en dos de las etapas principales que son el ADC y la DAC, en la Figura **??** se presenta la FSM principal que conforma todo el sistema del PID *wavenet*-difuso.

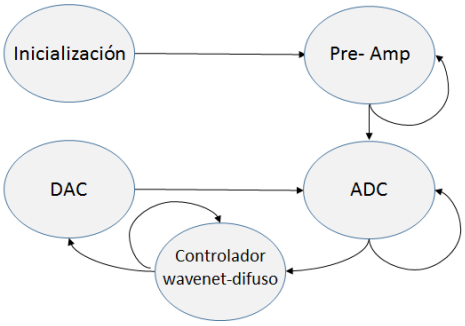

Figura 4.1: Máquina de estados finitos FSM principal.

En la primer etapa se inicializa el sistema asignando todos los valores iniciales de $\mathbf{W}(k), \mathbf{A}(k), \mathbf{B}(k), \mathbf{C}(k), \mathbf{D}(k)$, así como las ganancias iniciales del PID. En la segunda etapa se configura el pre-amplificador del ADC, posteriormente en la tercer etapa se hace la conversión análoga digital de la lectura obtenida del sensor de la planta. En la tercer etapa se encuentra el PID *wavenet*-difuso, en esta etapa se hace la estimación de la planta se calculan los valores asociados al PID, se calcula la señal de control y se actualizan todos los parámetros. Finalmente en la última etapa se hace la conversión digital-análoga de la señal de control obtenida por el controlador.

4.2. Adquisición de datos

Durante el proceso de adquisición de datos provenientes del sensor se ven involucradas dos etapas, estas etapas son el pre-amplificador y el ADC, ambos componentes se encuentran dentro de la tarjeta del *Spartan 3E Starter Kit* y son manufacturados por LINEAR TECHNOLOGY LTD. El chip correspondiente al ADC es el LTC1407A, este representa la señal análoga en una representación digital en complemento 2. El pre-amplificador es un chip independiente que tiene como propósito escalar el voltaje recibido en las entradas $V_{in}A$ y $V_{in}B$ para maximizar de esta forma el rango de conversión, este chip es el LTC6912-1. Para lograr la adquisición y conversión de los datos recibidos se diseña una FSM simple en VHDL, esta FSM tiene como propósito llevar al pre-amplificador y al ADC de un estado a otro, controlando los tiempos de algunas señales necesarias para el pre-amplificador y el ADC y capturando la salida del ADC basándose en el reloj del bus SPI, el cual es utilizado para leer las señales análogas de entrada y muestrear la salida.

4.2.1. Convertidor analógico - digital

Existen diversos tipos de ADCs (Convertidor Analógico Digital), el que se utilizara para esta tesis será el ADC de Aproximaciones Sucesivas (SAR-ADC), los principales

de este ADC incluyen una DAC (Convertidor Digital Analógico), un reloj, un comparador y un registro SAR para guardar los valores digitales que llegan una vez que el comparador compara los valores de la DAC con las entradas y salidas analógicas. Esta estructura la podemos apreciar en la Figura **??**.

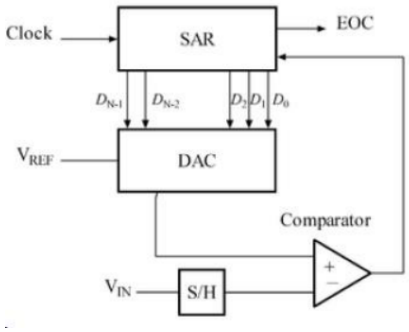

Figura 4.2: Diagrama a bloques del SAR-ADC.

Asi mismo en las Figuras **??** y **??** podemos observar el circuito del LTC1407A y el diagrama del circuito de captura analógica del Spartan 3E

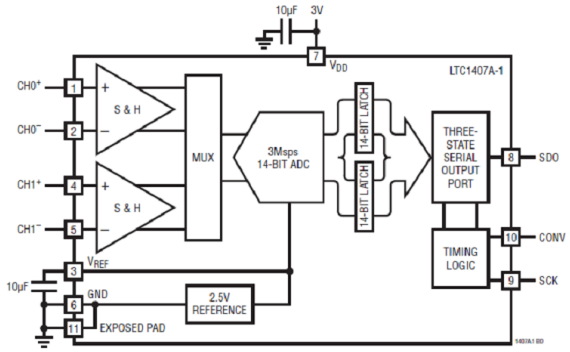

Figura 4.3: Circuito del LTC1407A.

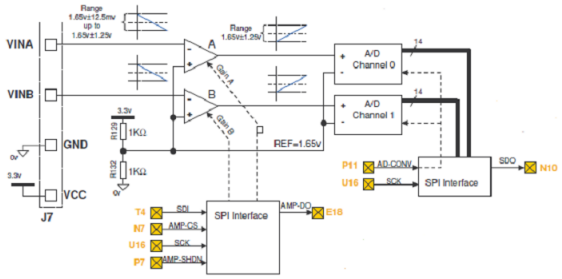

Figura 4.4: Circuito de captura analógica.

4.2.2. Pre-amplificador

El LTC6912-1 provee dos amplificadores inversores independientes con ganancia programable. El propósito de este amplificador es escalar el voltaje recibido en $V_{in}A$ y $V_{in}B$ para maximizar el rango de conversión, es decir $1.65 \pm 1.25V$. En la Figura **??** podemos observar el diagrama de dicho amplificador.

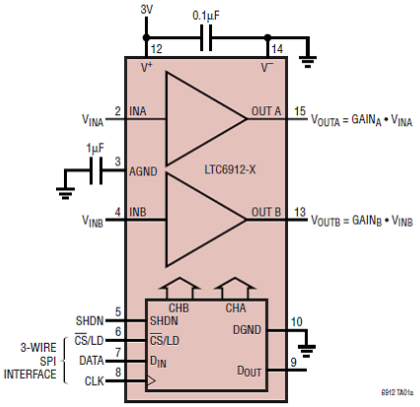

Figura 4.5: Diagrama del LTC6912-1.

Las ganancias de ambos canales pueden ser programadas de manera independiente por medio de una interfaz SPI para seleccionar ganancias de voltaje de 0, -1, -2, -5, -10, -20, -50 y -100.

4.2.3. Funcionamiento

Existen varias señales que son utilizadas para realizar la interfaz entre el pre-amplificador y el FPGA. El amplificador utiliza un bus SPI para comunicare con el FPGA, este bus SPI entre una variedad de dispositivos periféricos diferentes, como el ADC y la DAC. Por ello las señales de interfaz de estos dispositivos deben ser desactivadas para asegurar que exista un interacción propia del bus SPI con el pre-amplificador. Algunas de las señales que controlan el funcionamiento del pre amplificador son

- **SPI_MOSI**: Esta señal se encuentra en el pin T4 del *Spartan 3E*. Esta señal es dirigida del FPGA al ADC. El termino MOSI proviene de Master Output, Slave Input. Esta señal es utilizada para llevar el vector de 8 bits con el que se programa la ganancia de cada amplificador. En base a esta ganancia pre-cargada se llevara a cabo la conversión del valor análogo a digital. También el rango de voltaje de entrada al ADC depende de esta configuración.

- **AMP_CS**: Esta señal, localiza en el pin N7 del *Spartan 3E*, es dirigida del FPGA al pre-amplificador. Es una señal de selección de chip. La ganancia del amplificador es configurada cuando esta señal se encuentra en estado alto.

- **SPI_SCK**: Esta señal va del FPGA al pre-amplificador y se encuentra en el pin U16 del *Spartan 3E*. Esta señal es básicamente una señal de reloj de la cual depende la programación del pre amplificador. La señal SPI_MOSI envía un bit en cada flanco ascendente del reloj SPI_SCK.

- **AMP_SHDN**: Esta señal es una señal de reset que se desactiva en cada flanco ascendente del reloj SPI_SCK. Se encuentra en el pin P7 del *Spartan 3E* y va del FPGA al pre amplificador.

- **AMP_DOUT**: Esta señal se envía del pre amplificador al FPGA, se encuentra en el pin E18 del *Spartan 3E*. Esta señal simplemente regresa la configuración de ganancia del pre-amplificador al FPGA.

La configuración de la ganancia se logra programando el pre-amplificador mediante la señal SPI_MOSI. La ganancia para cada amplificador es enviada dentro de una palabra de 8 bits, que consiste en 2 campos de 4 bits cada uno como se puede apreciar en la Figura **??**.

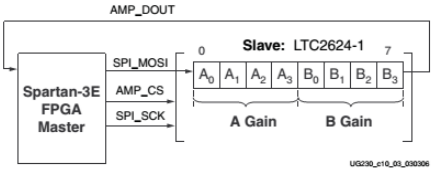

Figura 4.6: Interfaz SPI con el pre-amplificador.

La transacción del bus SPI inicia cuando el FPGA confirma que La señal AMP_CS se encuentra en estado bajo, el amplificador captura la información en SPI_MOSI un bit a la vez en cada flanco ascendente de la señal de reloj SPI_SCK. La información es enviada empezando por el bit más significativo (MSB), es decir, el bit B3 se envía primero y después los demás hasta enviar los 8 bits. En la Tabla **??** se muestran las configuraciones y sus respectivas ganancias permisibles para el pre-amplificador. El diagrama de tiempos se muestra en la Figura **??**.

Ganancia	A3	A2	A1	A0	Voltaje de entrada	
	B3	B2	B1	B0	Mínimo	Máximo
0	0	0	0	0		
-1	0	0	0	1	0.4	2.9
-2	0	0	1	0	1.025	2.275
-5	0	0	1	1	1.4	1.9
-10	0	1	0	0	1.525	1.775
-20	0	1	0	1	1.5875	1.7125
-50	0	1	1	0	1.625	1.675
-100	0	1	1	1	1.6375	1.6625

Tabla 4.1: Ganancias del pre-amplificador.

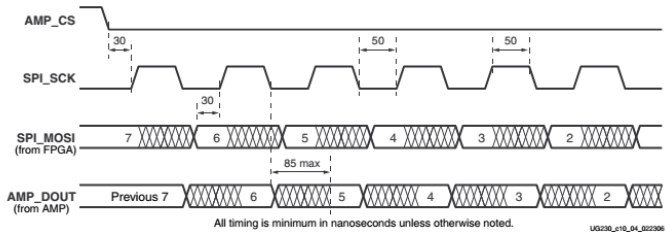

Figura 4.7: Tiempos de comunicacion del SPI y el pre-amplificador.

El ADC genera una salida digital de 14 bits a complemento 2 de la entrada analógica. El voltaje de entrada recibido en el ADC depende de la ganancia previamente programada en el pre-amplificador. El rango máximo que soporta la entrada del ADC se obtiene con la ganancia de -1, la cual nos permite un rango de $0.4V$ a $2.9V$. La conversión analógica–digital es dada por[hacer referencia al manual de xilinx]:

$$D[13:0] = GANANCIA \times \frac{V_{in} - 1.65V}{1.25V} \times 8192 \tag{4.1}$$

donde $D[13:0]$ representa el valor de 14 bits a complemento 2 de la entrada analógica, 1.65 es el voltaje de referencia del ADC la cual se obtiene dividiendo V_{cc} el cual es $3.3V$. Existen 3 señales adicionales que se utilizan para comunicar al ADC con el FPGA, dichas señales son:

- **AD_CONV**: Esta es una señal interna del FPGA, su función es marcar el inicio de la conversión de la señal analógica, se encuentra en el pin P11 y es enviada del FPGA al ADC.
- **SPI_MISO**: Esta señal es la salida serial del ADC al FPGA. Es la encargada de entregar la representación binaria de 14 bits a complemento dos de la señal

analógica muestreada. Esta señal se encuentra en el pin N10 y va del FPGA al ADC.

- **SPI_SCK**: Al igual que en el pre-amplificador, esta señal de reloj es sumamente importante en el proceso de conversión y también al enviar la información del ADC al FPGA.

Los dos resultados de las conversiones son enviados en 32 ciclos de reloj. Dos resultados de 14 bits son obtenidos del ADC, cabe mencionar que la señal AD_CONV no es una señal SPI tradicional, deben tomarse precauciones y asegurar de proveer suficientes ciclos de reloj del SPI_SCK para que el ADC deje la señal SPI_MISO en un estado de alta impedancia, de lo contrario, el ADC bloqueara la comunicación de los otros periféricos SPI. El ADC requiere de dos ciclos de reloj antes y después de cada transferencia de 14 bits (haciendo un total de 34 ciclos de reloj). Es importante recordar que las señales del bus SPI son compartidas con otros dispositivos del FPGA, es de vital importancia que otros dispositivos que no estén siendo utilizados sean desactivados cuando el FPGA se está comunicando con el ADC, el pre-amplificador o la DAC para evitar fallos de comunicación. Cuando la señal AD_CONV se pone en alto, el ADC de forma simultanea muestrea ambos canales análogos. El resultado de esta conversión no es mostrado hasta la próxima vez que la señal AD_CONV se haga presente en el bus SPI, creando por lo tanto, una latencia de un ciclo de reloj. La máxima velocidad de muestreo es de aproximadamente 1.5MHz. El diagrama de tiempo se muestra en la Figura **??**.

Ahora bien, la salida binaria de 14 bits satisface a la ecuación (4.1), pero como se puede observar, debido al voltaje de referencia interno de 1.65V del pre-amplificador, la salida digital obtenida no es directamente proporcional a la entrada analógica, sino que tiene un offset. El remover este offset hace que esta salida digital sea más conveniente de usar. El offset puede ser removido de la siguiente manera:

$$D = \frac{GAIN * V_{in}}{1.25} - \frac{GAIN * 1.65 * 8192}{1.25} \tag{4.2}$$

$$\Rightarrow D + \frac{GAIN * 1.65 * 8192}{1.25} = \frac{GAIN * V_{in}}{1.25} \tag{4.3}$$

ahora, si la ganancia es de -1 tendriamos que

$$D - 10813 = \frac{GAIN * V_{in}}{1.25} \tag{4.4}$$

convirtiendolo en formato de complemento 2

$$D[13:0] + 101010111000011 = \frac{GAIN * V_{in}}{1.25}$$

$$\Rightarrow D'[13:0] = \frac{GAIN * V_{in}}{1.25} \tag{4.5}$$

donde D' es directamente proporcional al voltaje análogo de entrada con el offset removido.

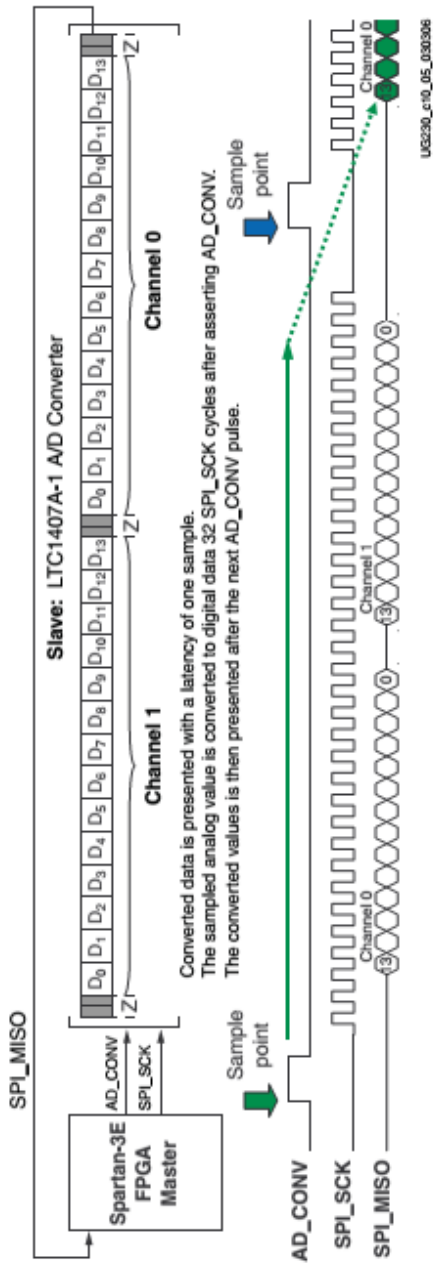

Figura 4.8: Tiempos de conversion del ADC.

4.2.4. Programación a VHDL

Una vez que entendemos el funcionamiento de estos dos componentes es posible diseñar una FSM para cada proceso. A continuación se presenta el programa en VHDL de la etapa de adquisición y conversión de la señal análoga, todo el proceso consta de un total de 2 FSM, una que controla la etapa del pre-amplificador y otra para el ADC. A la salida de la FSM del ADC tendremos un vector 32 bits con signo el cual representara la salida de nuestro sistema $y(k)$.

VHDL 4.2: Código VHDL de la FSM pre-amp

```
- - -Ciclo del pre-amp- - -
    when amp =>
       DAC_CS <='1';
       case amp_reg is
- - - idle - - -
    when idle =>
       contador_g := 7;
          if (enable = '1') then
             sck_reg <= '0';
             AMP_SHDN <= '0';
             amp_reg <= inicio;
          else
             AMP_SHDN <= '1';
             AMP_CS <= '1';
             amp_reg <= idle;
          end if;
- - - inicio - - - -
    when inicio =>
       AMP_SHDN <='0';
       sck_reg <='0';
       SPI_MOSI <='0';
       contador := 0 ;
       AMP_CS <='0';
          if (dummy_counter=1) then
             amp_reg <= hi;
             dummy_counter := 0;
          else
             dummy_counter := dummy_counter + 1;
             amp_reg <= inicio;
          end if;
```

```
- - - hi - - -
    when hi =>
          if (dummy_counter=1) then
             sck_reg <= '1';
             contador := contador+1;
             amp_reg <= dummy_lo;
             dummy_counter := 0;
          else
             SPI_MOSI <= ganancia(contador_g);
             dummy_counter := dummy_counter+1;
             amp_reg <= hi;
             end if;
- - - dummy_lo - - -
    when dummy_lo =>
          if (contador=8) then
             amp_reg <= lo;
          else
             sck_reg <='1';
             amp_reg <= lo;
          end if;
- - - lo - - -
    when lo =>
          if contador = 8 then
             AMP_SHDN <= '0';
             amp_reg <=nada;
             SPI_MOSI <= '0';
             AMP_CS <= '1';
             contador_g := 7;
             sck_reg <= '0';
             estado <= adc;
          else
             sck_reg <= '0';
             contador_g :=contador_g-1;
             amp_reg <= hi;
             AMP_CS <= '0';
          end if;
    when nada =>
       AMP_CS <='1';
    when others =>
       amp_reg <= idle;
    end case;
```

VHDL 4.3: Código VHDL de la FSM ADC

```
- - - Ciclo ADC - - -
    when adc =>
        AMP_CS <='1';
        DAC_CS <='1';
        case adc_reg is
- - - idle_adc - - -
    when idle_adc =>
        contador := 0;
        sck_reg <='0';
            if (enable = '0') then
                AD_CONV <='0';
                AMP_SHDN <='1';
                estado <= amp;
                amp_reg <= idle;
            else
                adc_reg <= inicio_adc;
                AD_CONV <='1';
            end if;
- - - inicio_adc - - -
    when inicio_adc =>
        sck_reg <='0';
        AD_CONV <='0';
        cuenta := 17;
        adc_reg <= lo_adc;
- - - lo_adc - - -
        when lo_adc =>
        sck_reg <= '0';
            if (cuenta>0) then
                cuenta := cuenta-1;
            end if;
        contador := contador+1;
        adc_reg <= hi_adc;
- - - hi_adc - - -
    when hi_adc =>
        sck_reg <= '1';
            if (contador <= 2) then
                adc_reg <= lo_adc;
```

```
            elsif contador >2 and contador <= 16 then
               ADC1(cuenta) <= SPI_MISO;
               adc_reg <= lo_adc;
            elsif contador = 17 then
               adc_reg <= lo_adc;
            else
               adc_reg <= fine_adc;
            end if;
- - - fine_adc - - -
      when fine_adc =>
         contador := 0;
         cuenta := 17;
         sck_reg <= '0';
         adc_reg <= correccion;
- - - correccion - - -
      when correccion =>
         ADC1_corregida <= errorfix +(ADC1(13) & ADC1);
         adc_reg <= idle_adc;
         estado <= wavenet;
      when others =>
         adc_reg <= idle_adc;
      end case;
```

4.3. Arquitectura del controlador

Al igual que la etapa de adquisición de datos, la arquitectura diseñada para el controlador PID *wavenet* es una simple FSM que sigue el algoritmo mostrado en la Tabla 3.3. El diseño de la FSM se muestra en la Figura **??**.

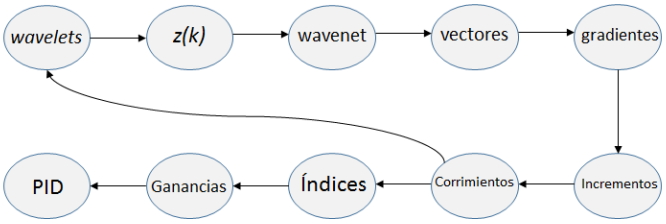

Figura 4.9: FSM del controlador.

Durante el primer estado de esta etapa se calculan las *wavelets* hijas y sus derivadas con respecto a $b(k)$, para esto necesario calcular primeramente el valor de τ_l. El segundo estado corresponde al cálculo de la salida $z(k)$, Ecuación (3.4). Posteriormente en el siguiente estado se calcula la salida de la *wavenet* con el filtro IIR $\hat{y}(k)$. El estado llamado "vectores" es donde se calculan los vectores Ψ_{b_l} y Ψ_b estos vectores son necesarios para poder calcular los gradientes que serán calculados en el estado siguiente. Una vez que los gradientes han sido calculados se procede a calcular los

incrementos y posteriormente los corrimientos, en este estado la FSM comprueba el número de épocas que han concluido, si ya se han completado las épocas establecidas para el aprendizaje procederá a pasar al estado donde se calculan las tasas de aprendizaje de las ganancias PID, de lo contrario regresara al inicio de la etapa para realizar todo el proceso nuevamente.

Una vez que los índices han sido calculados, por medio de las Ecuaciones (3.35) (3.36) y (3.37), se calculan las ganancias del controlador PID. Por último se calcula la ley de control $u(k)$. A continuación se presenta el programa en VHDL de la etapa del controlador.

VHDL 4.4: Código en VHDL del controlador

```
when wavenet =>
case wavenet_reg is
when taos =>
k <= to_sfixed (k_int, k);
tao1 <= resize ( (k - b1)/a1, tao1'high, tao1' low);
tao2 <= resize ( (k - b2)/a2, tao2'high, tao2' low);
tao3 <= resize ( (k - b3)/a3, tao3'high, tao3' low);
wavenet_reg <= wavelets;
when wavelets =>
m_wavelet1 <= resize ( tao1/(((tao1*tao1)+1)*((tao1*tao1)+1)), m_wavelet1'
high, m_wavelet1' low);
m_wavelet2 <= resize ( tao2/(((tao2*tao2)+1)*((tao2*tao2)+1)), m_wavelet2'
high, m_wavelet2' low);
m_wavelet3 <= resize ( tao3/(((tao3*tao3)+1)*((tao3*tao3)+1)), m_wavelet3'
high, m_wavelet3' low);
z <= resize ((w1*m_wavelet1)+(w2*m_wavelet2)+(w3*m_wavelet3), z' high, z'
low);
wavenet_reg <= iir;
when iir =>
gama <= resize ((z*c0)+(z_1*c1)+(z_2*c2)*u, gama' high, gama' low);
fi <= resize((y_1*c1)+(y_2*c2)*v, fi' high, fi' low);
y <= resize (gama+fi, y' high, y' low);
y_t <= to_sfixed(ADC1_corregida, y_t);
e_err <= resize ((y_t - y), e_err' high, e_err' low);
wavenet_reg <= vectores;
when vectores =>
dev_wavb1 <= resize (reciprocal(a1)*((3*(tao1*tao1)-
1)/(((tao1*tao1)+1)*((tao1*tao1)+1)*((tao1*tao1)+1))),dev_wavb1' high,
dev_wavb1' low);
```

```
dev_wavb1_1 <= resize (reciprocal(a1)*((3*(tao1_1*tao1_1)-
1)/(((tao1_1*tao1_1)+1)*((tao1_1*tao1_1)+1)*((tao1_1*tao1_1)+1))),
dev_wavb1_1'high, dev_wavb1_1' low);
dev_wavb1_2 <= resize (reciprocal(a1)*((3*(tao1_2*tao1_2)-
1)/(((tao1_2*tao1_2)+1)*((tao1_2*tao1_2)+1)*((tao1_2*tao1_2)+1))),
dev_wavb1_2'high, dev_wavb1_2' low);
dev_wavb2 <= resize (reciprocal(a2)*((3*(tao2*tao2)-
1)/(((tao2*tao2)+1)*((tao2*tao2)+1)*((tao2*tao2)+1))), dev_wavb2' high,
dev_wavb2' low);
dev_wavb2_1 <= resize (reciprocal(a2)*((3*(tao2_1*tao2_1)-
1)/(((tao2_1*tao2_1)+1)*((tao2_1*tao2_1)+1)*((tao2_1*tao2_1)+1))),
dev_wavb2_1'high, dev_wavb2_1' low);
dev_wavb2_2 <= resize (reciprocal(a2)*((3*(tao2_2*tao2_2)-
1)/(((tao2_2*tao2_2)+1)*((tao2_2*tao2_2)+1)*((tao2_2*tao2_2)+1))),
dev_wavb2_2'high, dev_wavb2_2' low);
dev_wavb3 <= resize (reciprocal(a3)*((3*(tao3*tao3)-
1)/(((tao3*tao3)+1)*((tao3*tao3)+1)*((tao3*tao3)+1))), dev_wavb3' high,
dev_wavb3' low);
dev_wavb3_1 <= resize (reciprocal(a3)*((3*(tao3_1*tao3_1)-
1)/(((tao3_1*tao3_1)+1)*((tao3_1*tao3_1)+1)*((tao3_1*tao3_1)+1))),
dev_wavb3_1' high, dev_wavb3_1' low);
dev_wavb3_2 <= resize (reciprocal(a3)*((3*(tao3_2*tao3_2)-
1)/(((tao3_2*tao3_2)+1)*((tao3_2*tao3_2)+1)*((tao3_2*tao3_2)+1))),
dev_wavb3_2' high, dev_wavb3_2' low);
wavenet_reg <= gradientes;
when gradientes =>
grad_ew1 <= resize (((-e_err*c0*m_wavelet1)-(e_err*c1*m_wavelet1_1)-
(e_err*c2*m_wavelet1_2))*u, grad_ew1' high, grad_ew1' low);
grad_ew2 <= resize (((-e_err*c0*m_wavelet2)-(e_err*c1*m_wavelet2_1)-
(e_err*c2*m_wavelet2_2))*u, grad_ew2' high, grad_ew2' low);
grad_ew3 <= resize (((-e_err*c0*m_wavelet3)-(e_err*c1*m_wavelet3_1)-
(e_err*c2*m_wavelet3_2))*u, grad_ew3' high, grad_ew3' low);
grad_eb1 <= resize (((-e_err*c0*dev_wavb1)-(e_err*c1*dev_wavb1_1)-
(e_err*c2*dev_wavb1_2))*w1*u,grad_eb1' high, grad_eb1' low);
grad_eb2 <= resize (((-e_err*c0*dev_wavb2)-(e_err*c1*dev_wavb2_1)-
(e_err*c2*dev_wavb2_2))*w2*u,grad_eb2' high, grad_eb2' low);
grad_eb3 <= resize (((-e_err*c0*dev_wavb3)-(e_err*c1*dev_wavb3_1)-
(e_err*c2*dev_wavb3_2))*w3*u,grad_eb3' high, grad_eb3' low);
grad_ea1 <= resize (tao1*grad_eb1, grad_ea1' high, grad_ea1' low);
grad_ea2 <= resize (tao2*grad_eb2, grad_ea2' high, grad_ea2' low);
grad_ea3 <= resize (tao3*grad_eb3, grad_ea3' high, grad_ea3' low);
```

```
grad_ec0 <= resize (-e_err*z*u, grad_ec0' high, grad_ec0' low);
grad_ec1 <= resize (-e_err*z_1*u, grad_ec1' high, grad_ec1' low);
grad_ec2 <= resize (-e_err*z_2*u, grad_ec2' high, grad_ec2' low);
grad_ed1 <= resize (-e_err*y_1*v, grad_ed1' high, grad_ed1' low);
grad_ed2 <= resize (-e_err*y_2*v, grad_ed2' high, grad_ed2' low);
wavenet_reg <= deltas;
when deltas =>
delta_w1 <= resize (-(grad_ew1), delta_w1' high, delta_w1' low);
delta_w2 <= resize (-(grad_ew2), delta_w2' high, delta_w2' low);
delta_w3 <= resize (-(grad_ew3), delta_w3' high, delta_w3' low);
delta_b1 <= resize (-(grad_eb1), delta_b1' high, delta_b1' low);
delta_b2 <= resize (-(grad_eb2), delta_b2' high, delta_b2' low);
delta_b3 <= resize (-(grad_eb3), delta_b3' high, delta_b3' low);
delta_a1 <= resize (-(grad_ea1), delta_a1' high, delta_a1' low);
delta_a2 <= resize (-(grad_ea2), delta_a2' high, delta_a2' low);
delta_a3 <= resize (-(grad_ea3), delta_a3' high, delta_a3' low);
delta_c0 <= resize (-(grad_ec0), delta_c0' high, delta_c0' low);
delta_c1 <= resize (-(grad_ec1), delta_c1' high, delta_c1' low);
delta_c2 <= resize (-(grad_ec2), delta_c2' high, delta_c2' low);
delta_d1 <= resize (-(grad_ed1), delta_d1' high, delta_d1' low);
delta_d2 <= resize (-(grad_ed2), delta_d2' high, delta_d2' low);
wavenet_reg <= incrementos;
when incrementos =>
new_w1 <= resize (w1+(mw*delta_w1), new_w1' high, new_w1' low);
new_w2 <= resize (w2+(mw*delta_w2), new_w2' high, new_w2' low);
new_w3 <= resize (w3+(mw*delta_w3), new_w3' high, new_w3' low);
new_b1 <= resize (b1+(mb*delta_b1), new_b1' high, new_b1' low);
new_b2 <= resize (b2+(mb*delta_b2), new_b2' high, new_b2' low);
new_b3 <= resize (b3+(mb*delta_b3), new_b3' high, new_b3' low);
new_a1 <= resize (a1+(ma*delta_a1), new_a1' high, new_a1' low);
new_a2 <= resize (a2+(ma*delta_a2), new_a2' high, new_a2' low);
new_a3 <= resize (a3+(ma*delta_a3), new_a3' high, new_a3' low);
new_c0 <= resize (c0+(mc*delta_c0), new_c0' high, new_c0' low);
new_c1 <= resize (c1+(mc*delta_c1), new_c1' high, new_c1' low);
new_c2 <= resize (c2+(mc*delta_c2), new_c2' high, new_c2' low);
new_d1 <= resize (d1+(md*delta_d1), new_d1' high, new_d1' low);
new_d2 <= resize (d2+(md*delta_d2), new_d2' high, new_d2' low);
wavenet_reg <= corrimientos;
when corrimientos =>
if (epoc = 19) then
epoc := 0;
wavenet_reg <= taos;
estado <= pid;
else
```

```
w1 <= new_w1;
w2 <= new_w2;
w3 <= new_w3;
a1 <= new_a1;
a2 <= new_a2;
a3 <= new_a3;
b1 <= new_b1;
b2 <= new_b2;
b3 <= new_b3;
c0 <= new_c0;
c1 <= new_c1;
c2 <= new_c2;
d1 <= new_d1;
d2 <= new_d2;
z_2 <= z_1;
z_1 <= z;
y_2 <= y_1;
y_1 <= y;
tao1_2 <= tao1_1;
tao1_1 <= tao1;
tao2_2 <= tao2_1;
tao2_1 <= tao2;
tao3_2 <= tao3_1;
tao3_1 <= tao3;
m_wavelet1_2 <= m_wavelet1_1;
m_wavelet1_1 <= m_wavelet1;
m_wavelet2_2 <= m_wavelet2_1;
m_wavelet2_1 <= m_wavelet2;
m_wavelet3_2 <= m_wavelet3_1;
m_wavelet3_1 <= m_wavelet3;
epoc := epoc+1;
wavenet_reg <= taos;
end if;
when others =>
estado <= dac;
end case;
```

4.4. Conversor digital análogo

El *Spartan 3E* incluye un convertidor digital análogo (DAC) serial de cuatro canales, compatible con el bus SPI. Este dispositivo es el LTC2624 de Linear Technology. El diagrama de este dispositivo es mostrado en la Figura **??**.

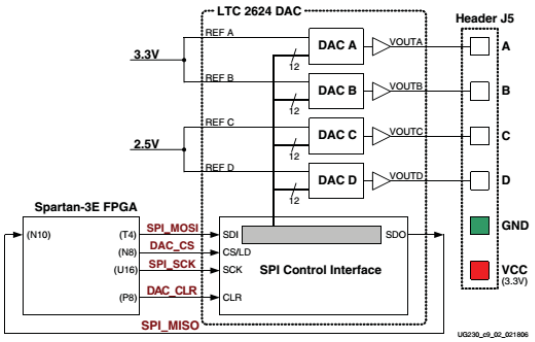

Figura 4.10: Esquematico de conexiones del dispositivo DAC.

Al igual que como se vio en la sección del ADC, la DAC comparte el mismo bus SPI y hay dos señales adicionales que se ven involucradas en este proceso, a continuación se muestran dichas señales.

- **DAC_CS**: Esta señal se encuentra en el pin N8, es dirigida del FPGA a la DAC, esta señal es de seleccion, la conversión es llevada a cabo cuando esta señal se pone en alto.

- **DAC_CLR**: Localizada en el pin P8, se dirige del FPGA a la DAC y es una entrada de reset asíncrona.

El funcionamiento de la DAC es muy similar al ADC, esta envía un bit de información conforme al SPI_SCK. En la Figura **??** se muestra el diagrama de tiempo de la DAC.

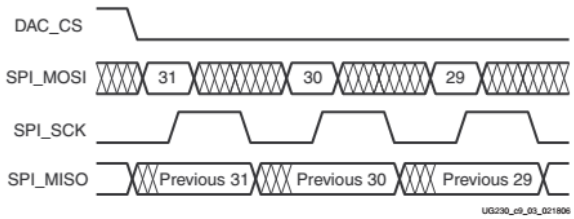

Figura 4.11: Diagrma de tiempo de la DAC.

Una vez que la señal DAC_CS es puesta en bajo, el FPGA transmite los datos a través del SPI_MOSI, empezando por el bit más significativo. El LTC2624 captura la información en cada flanco ascendente del SPI_SCK . Es muy importante que la información permanezca en un estado valido al menos por $4ns$ relativos al flanco ascendente del reloj.

La DAC transmite sus datos en la señal SPI_MISO en cada flanco descendente del SPI_SCK. El FPGA percibe esta información en el próximo flanco ascendente del SPI_SCK. Es importante notar que el FPGA debe leer el primer valor del SPI_MISO en el primer flanco ascendente del SPI_SCK que se presenta después de poner la señal DAC_CS en bajo, de lo contrario, se perderá el bit 31. Dentro del convertidor digital análogo, la interface SPI está formada por un shift register de 32 bits. Cada plabra de 32 bits consiste en un comando, una dirección, seguidos de los datos del valor a convertir, esto se puede observar en la Figura **??**. El comando más comúnmente utilizado es el COMMAND$[3:0]$ = "0011", este comando actualiza de forma inmediata la salida de la DAC seleccionada con el valor asignado.

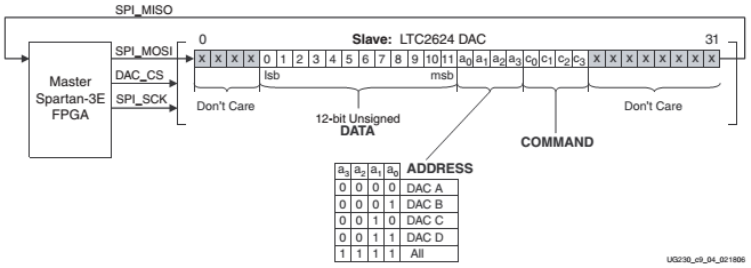

Figura 4.12: Transmisión de datos del LTC2624.

El nivel de salida en cada canal de la DAC es el equivalente análogo de un valor digital de 12 bits, $D[11:0]$. El voltaje en una salida especifica es dado por:

$$V_{out} = \frac{D[11:0]}{4096} \times V_{ref} \tag{4.6}$$

donde V_{ref} es diferente entre las salidas de la DAC. Los canales A y B utilizan un voltaje de referencia de $3.3V$ mientras que los canales C y D usan un voltaje de referencia de $2.5V$. Es importante recordar que los voltajes de referencia tienen una tolerancia de $\pm 5V$ por lo tanto para los canales A y B tendríamos que:

$$V_{out} = \frac{D[11:0]}{4096} \times (3.3V \pm 5\,\%) \tag{4.7}$$

y para los canales C y D tendríamos que:

$$V_{out} = \frac{D[11:0]}{4096} \times (2.5V \pm 5\,\%) \tag{4.8}$$

Capítulo 5

Resultados

En el presente capítulo se muestran los resultados simulados y experimentales obtenidos al aplicar el controlador PID *wavenet*-difuso, el cual fue descrito en el Capítulo 3, para controlar la posición angular del carro péndulo invertido, así como la velocidad angular de un motor de CD. La simulación se hace con el objetivo de validar el funcionamiento del controlador así como la arquitectura del mismo, para después programarlo en un FPGA de la familia *Spartan 3E*.

Este capítulo se encuentra organizado de la siguiente manera: en la Sección 5.1 se da una breve introducción que describe las plataformas utilizadas. En la Sección 5.2 se presentan los resultados de simulación del carro péndulo invertido y en la Sección 5.3 se presentan los resultados experimentales del motor de CD. Por ultimo en la Sección 5.4 se muestran algunos comentarios de este capítulo.

5.1. Introducción

MatLab es una herramienta de software matemático ampliamente utilizado para simulaciones numéricas, esto debido a que ofrece muchas prestaciones muy útiles, como son, la manipulación de matrices, representación de datos y funciones, implementación de algoritmos, creación de interfaces de usuario (GUI) así como la comunicación con programas en otros lenguajes y dispositivos de hardware. MatLab dispone de una herramienta adicional que expande estas prestaciones, esta herramienta es *Simulink*, *Simulink* es un entorno de programación visual que nos ofrece un nivel más alto de abstracción que el lenguaje interpretado por MatLab. *Simulink* se ha convertido en una herramienta muy utilizada en ingeniería de control y robótica ya que es una poderosa herramienta de simulación de modelos o sistemas, con cierto grado de abstracción de los fenómenos físicos involucrados en los mismos. Los resultados de simulación que son presentados en esta sección son hechos en *Simulink* al igual que los modelos utilizados para este objetivo.

5.2. Simulación carro péndulo invertido

La primera planta en la cual se aplica el controlador PID *wavenet*-difuso es el sistema carro péndulo invertido con el fin de controlar su posición angular. Para lograr la simulación se considera el siguiente par de ecuaciones [**?**]

$$(M+m)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = u \tag{5.1}$$
$$(ml^2+J)\ddot{\theta} + ml\ddot{x}\cos\theta - mgl\,\mathrm{sen}\,\theta = 0 \tag{5.2}$$

este sistema de ecuaciones describe el comportamiento del carro péndulo invertido, ahora bien utilizando las variables de estado

$$x_1 = x, \quad x_2 = \theta, \quad x_3 = \dot{x}, \quad x_4 = \dot{\theta} \tag{5.3}$$

el sistema (5.1)-(5.2) se puede representar mediante un sistema de ecuaciones de la forma

$$\dot{x} = f(x) + g(x)u \tag{5.4}$$

donde $x, \dot{x}, \ddot{x} \in \mathbb{R}^4$, $u \in \mathbb{R}$ y

$$f(x) = \begin{bmatrix} x_3 \\ x_4 \\ \frac{m^2gl^2\sin x_2\cos x_2 - (ml^2+J)mlx_4^2\sin x_2}{-(M+m)(ml^2+J)+m^2l^2\cos^2 x_2} \\ \frac{-mgl(M+m)\sin x_2 + m^2l^2x_4^2\sin x_2\cos x_2}{-(M+m)(ml^2+J)+m^2l^2\cos^2 x_2} \end{bmatrix} \tag{5.5}$$

$$g(x) = \begin{bmatrix} 0 \\ 0 \\ \frac{-(ml^2+J)}{-(M+m)(ml^2+J)+m^2l^2\cos^2 x_2} \\ \frac{ml\cos x_2}{-(M+m)(ml^2+J)+m^2l^2\cos^2 x_2} \end{bmatrix} \tag{5.6}$$

Los parámetros de simulación son mostrados en la Tabla **??**, en la Tabla **??** se muestran los parámetros del controlador PID *wavenet*-difuso y en la Tabla **??** los parámetros y variables del sistema carro péndulo invertido.

Tiempo de simulación	$15seg$
Tiempo de muestreo	$0.01seg$

Tabla 5.1: Parámetros de simulación.

PARAMETRO	VALOR INICIAL	PARAMETRO	VALOR INICIAL
W	$[3.2, -4, -3]$	μ_w	0.01
A	$[-120, -4.5, -12.2]$	μ_a	0.01
B	$[92.5, 29.5, 106]$	μ_b	0.01
C	$[0.025, -1.5, -1]$	μ_c	0.01
D	$[0.43, 1.75]$	μ_d	0.01
p	30	μ_p	0.3
i	1	μ_i	0.001
d	3	μ_d	0.08

Tabla 5.2: Parámetros del controlador.

Variables	Nombre	Valor	Unidades
x	Posición del carro		m
θ	Posición angular del péndulo		rad
Parámetro			
l	Longitud del péndulo	0.32	m
m	Masa del péndulo	0.23	kg
M	Masa del carro	0.52	kg
g	Constante de gravedad	9.81	m/seg^2
J	Inercia de la barra	0.007	$Kg \cdot m^2$

Tabla 5.3: Parámetros del sistema.

En la Figura **??** se muestra el diagrama diseñado en Simulink para propósito de simulación del controlador.

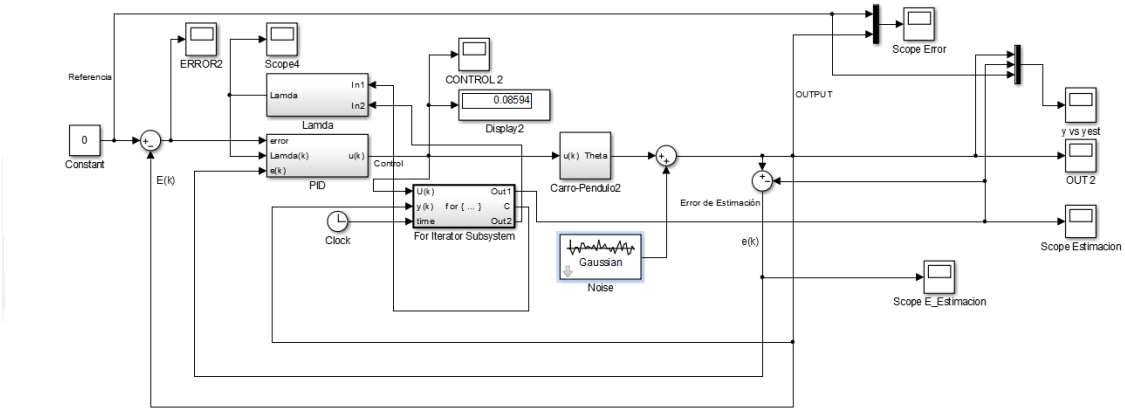

Figura 5.1: Diseño del controlador en Simulink.

A continuación en la Figura **??** se muestra el cojunto de reglas que fueron establecidas en Simulink para el sistema difuso que sintoniza de forma automatica los índices de aprendizaje del controlador PID.

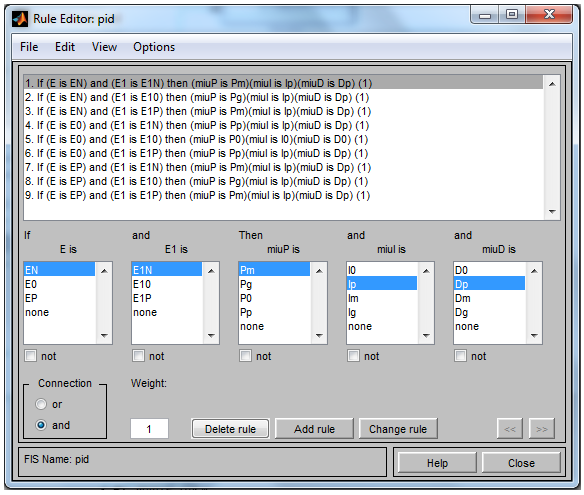

Figura 5.2: Conjunto de reglas difusas.

En la Figura **??** se muestra la interfaz de las funciones de membresía de Simulink, aqui en esta sección se crean los conjuntos difusos y las diferentes funciones de membresía.

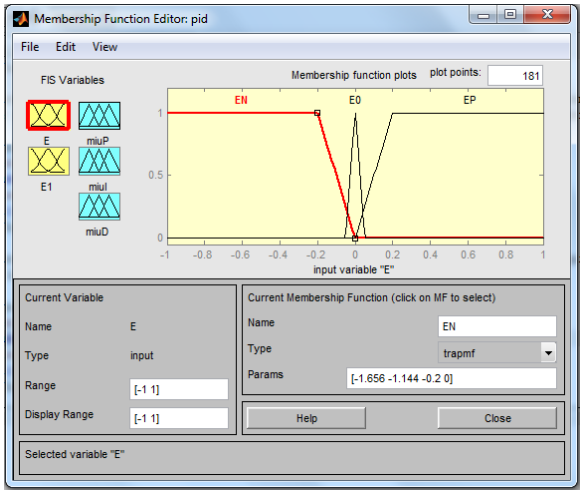

Figura 5.3: Interfaz para manipulación de funciones de membresía.

Ahora bien, una vez que se realiza la simulación podemos observar el comportamiento de los diversos parámetros de la wavenet, asi como los parámetros del PID. En la Figura **??** se presenta el comportamiento de los parámetros $A(k)$ y $B(k)$.

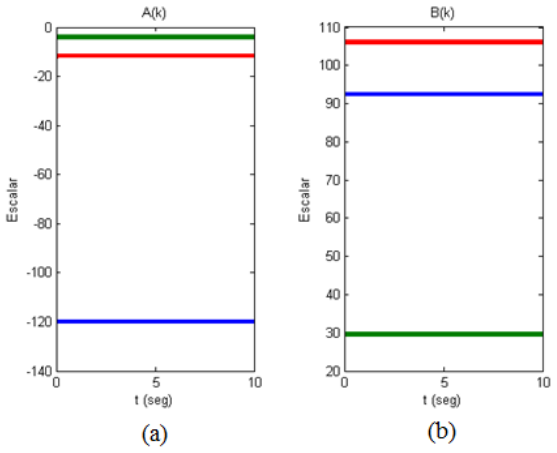

Figura 5.4: Comportamiento de los parámetros $A(k)$ y $B(k)$.

Los parámetro $C(k)$ y $D(k)$ se muestran en la Figura **??**.

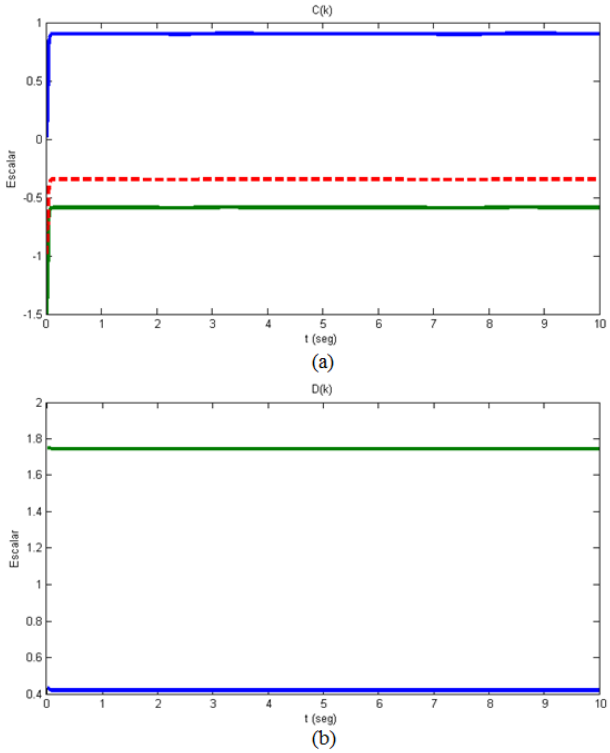

Figura 5.5: Comportamiento de los parámetros $C(k)$ y $D(k)$.

En la Figura **??** se muestra la evolución del parámetro $W(k)$

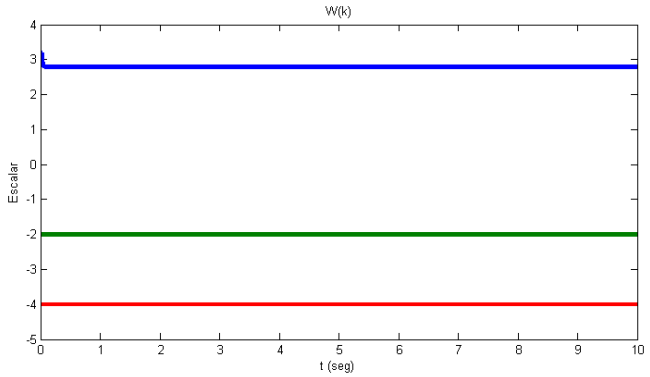

Figura 5.6: Comportamiento del parámetro $W(k)$.

En la Figura **??** se observa el comportamiento de las ganancias k_p, k_i y k_d del controlador PID.

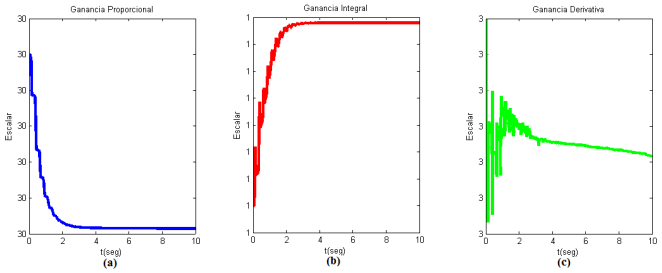

Figura 5.7: Evolución de los parámetros k_p(a), k_i(b) y k_d(c).

En la Figura **??** se muestra el comportamiento de los índices de aprendizaje del controlador PID.

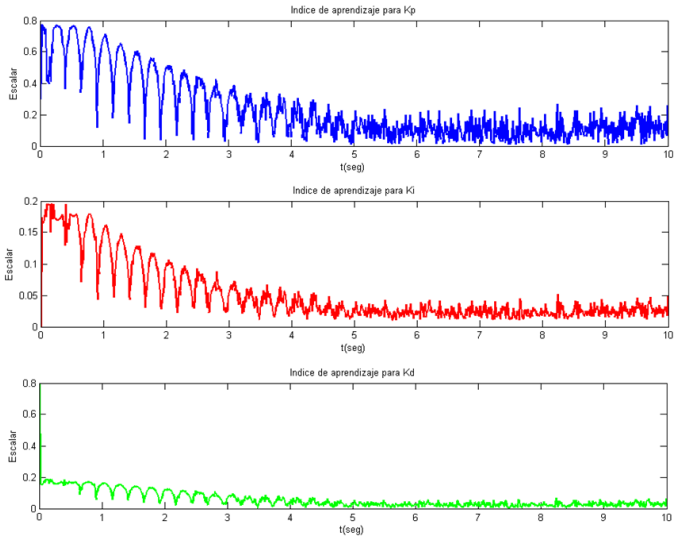

Figura 5.8: Evolución de los parámetros μ_p, μ_i y μ_d.

Por último en las Figuras **??** y **??** se muestra el comportamiento de la planta y la señal de control.

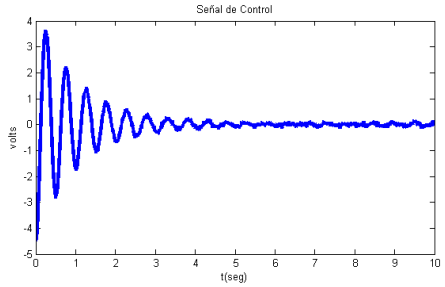

Figura 5.9: Salida de la planta.

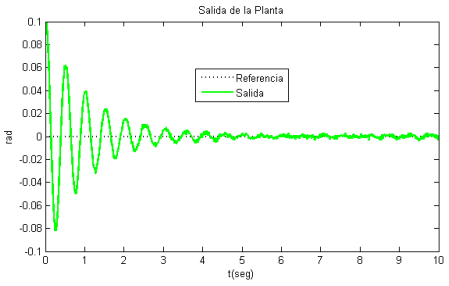

Figura 5.10: Señal de control.

Capítulo 6

Implementación en un FPGA

En este capítulo se presentan los resultados obtenidos de la implementación del controlador PID wavenet-difuso basado en una red neuronal wavelet cuyas funciones de activación son wavelets hijas RASP1. El capítulo está formado por dos secciones, en la Sección 6.1 se da una breve introducción del FPGA Spartan 3E para describir los puertos que se utilizaran finalmente en la Sección 6.2 se presenta el esquema de la plataforma experimental y los resultados obtenidos.

6.1. FPGA Spartan 3E

El Spartan 3E starter kit de Xilinx es una tarjeta de desarrollo que ofrece una gran versatilidad y que puede ser usado para prototipos de todo tipo ya que cuenta con componentes y características muy útiles en la aplicaciones embebidas. A continuación se mencionan algunos de los componentes del Spartan 3E, mientras que en la Figura **??** se muestra el FPGA Spartan 3E.

- Más de 10,000 celdas lógicas
- 232 pines I/O
- DDR SDRAM de 64Mbytes
- Flash SPI serial de 16 Mbits
- Pantalla LCD de 2 líneas, 16 caractéres
- Puerto VGA
- Puertos RS-232 de 9 pines
- Reloj oscilador interno de 50MHz

- Conversor análogo digital basado en SPI de dos entradas con amplificador programable
- Conversor digital análogo basado en SPI de cuatro salidas
- Encoder rotativo con push button

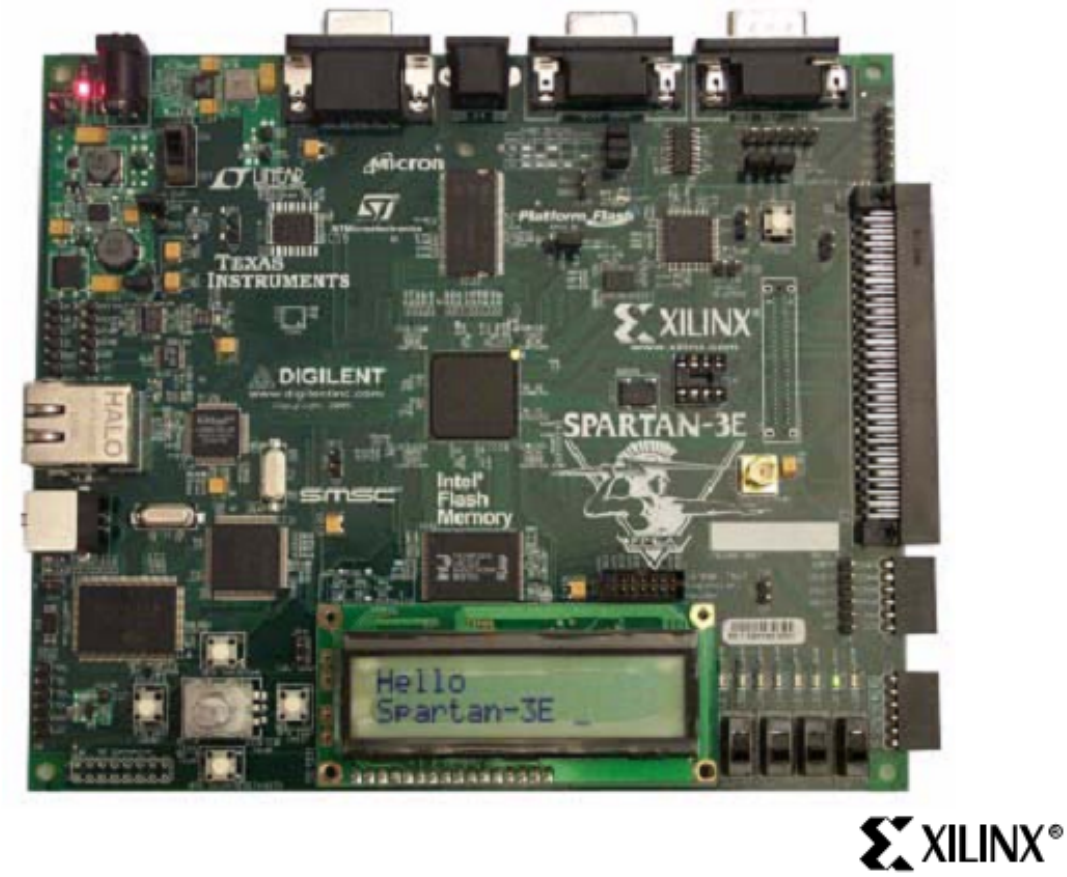

Figura 6.1: FPGA Spartan 3E.

6.2. Resultados obtenidos

La plataforma experimental se muestra en la Figura **??**, es importante observar que la tarjeta de adquisicion de datos solo se utiliza para obtener la grafica de respuesta de la planta que se mostrara posteriormente.

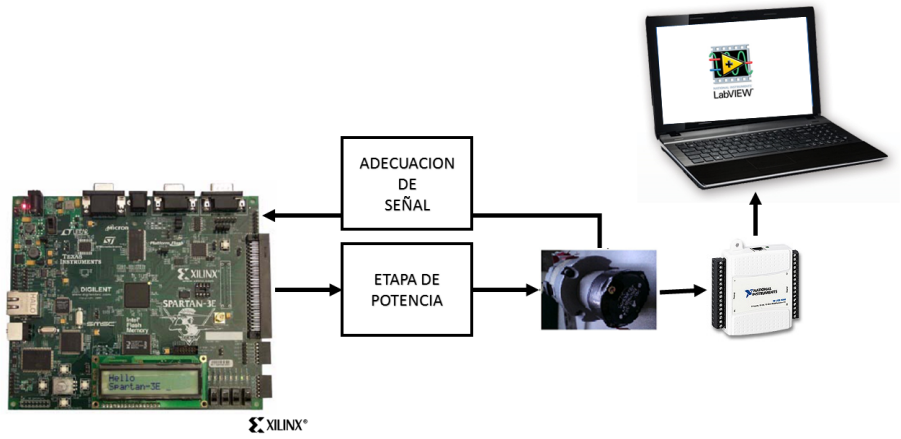

Figura 6.2: Diagrama a bloques de la plataforma experimental.

Como ya se comentó en el Capítulo 4, el Spartan 3E solo soporta un voltaje de entrada máximo de 2.9V, por esta razón es importante tener el circuito de adecuación de señal para evitar daños en el dispositivo, de igual forma la etapa de potencia es importante para lograr arrancar el motor, para esto es suficiente la implementación de un puente H y de un amplificador si se utiliza el voltaje directamente ó un mosfet si se desea utilizar PWM como señal de control. En la Figura **??** se muestra una imagen de la plataforma experimental.

Figura 6.3: Plataforma experimental.

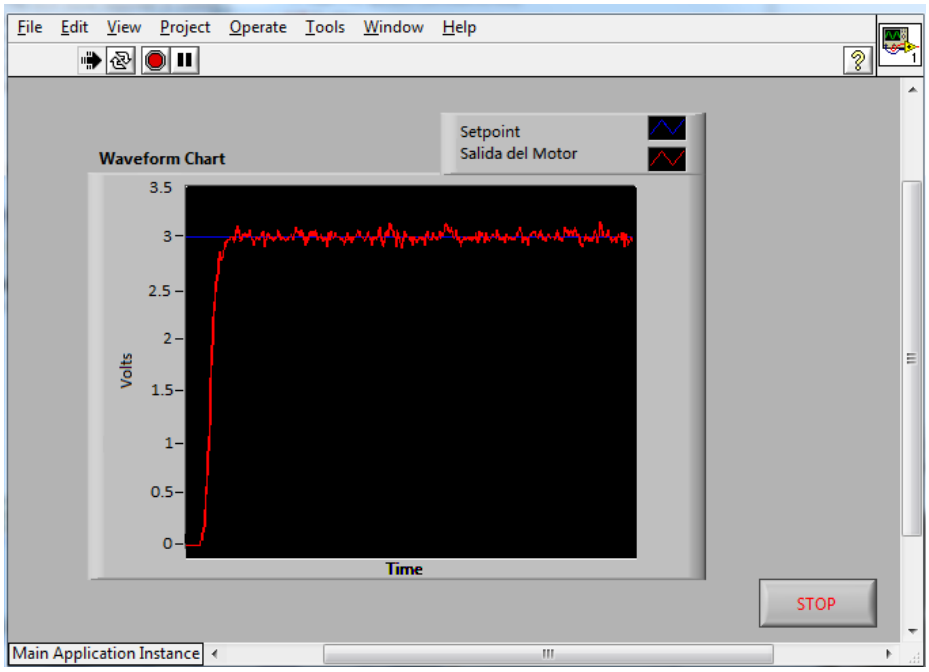

Figura 6.4: Resultado obtenido al Implementar el código en el FPGA.

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

En este trabajo de investigación se implemento un PID wavenet difuso el cual sintoniza en línea las ganancias proporcional, integral y derivativa de un controlador PID clásico discreto, el controlador fue simulado en Simulink para controlar un carro péndulo invertido y se implemento en un FPGA Spartan 3E para controlar la velocidad de un motor de CD, eliminando asi la dependencia de softwares y tarjetas de adquisicion de datos adicionales al FPGA, presentando buenos resultados en ambos casos. Se realizó un estudio comparativo entre el controlador seleccionado contra un PID clásico y un PID wavenet, obteniendo asi un mejor punto de vista de la diferencia que existe en cuanto a rendimiento entre dichos controladores, de lo anterior se concluye que el objetivo del trabajo de tesis se cumplió satisfactoriamente

7.2. Trabajos futuros

Algunos de los trabajos futuros que se desprenden de esta tesis son:

- Implementar un algoritmo que sea capaz de adaptarse a las diversas plantas cambiando la base de reglas difusas durante el ciclo de operación.
- Realizar un riguroso análisis de la convergencia y estabilidad en lazo cerrado de los nuevos algoritmos que incluyen redes neuronales y logica difusa.

Bibliografía

[1] A. Abu-Khudhair, R. Muresan, S.X. Yang, *Fuzzy control of semi-active automotive suspensions*, inProc. of IEEE Int. Conf. on Machatronics and Automation, vol. 54, no. 4, pp. 1937-1945, 2009.

[2] Eduardo Steed Espinoza Quesada, *Control de un sistema subactuado por visión artificial basado en un DSP*, Tesis de maestría, Universidad Autónoma del Estado de Hidalgo, 2006.

[3] Aws Abu-Khudhair, Radu Muresan, Simon X. Yang, *FPGA Based Real-Time Adaptive Fuzzy Logic Controller*, Proceedings of the 2010 IEEE International Conference on Automation and Logistics, Hong Kong and Macau, 2010.

[4] K. Aström. *Computer-Controlled Systems, Theory and Design.* Pretince-Hall, New York, 1997.

[5] K. J. Astrom, Tore Hagglund. *PID Controllers: Theory, Design and Tuning.* Internacional, 2007.

[6] J. A. Anderson. *A simple neural network generating and interactive memory.* Mathematical Biosciences, Vol. 14, pp. 197-220, 1972.

[7] V.S. Bagad, *VLSI Design*, Technical Publications Pune. Second revised edition, 2009.

[8] M. D. Baldania., D. A. Sawant., A. B. Patki, *DYNAMIC RULE BASED APPROACH TO REDUCE POWER CONSUMPTION OF THE FUZZY LOGIC CONTROLLER FOR EMBEDDED APPLICATIONS*, First International Conference on Networks & Soft Computing, 2014.

[9] C. M. Chang and T. S. Lui. *A Wavelet Network Control Method for Disk Drives.* IEEE Transactions on Control Systems Technology, Vol. 14, num. 1, pp. 63-68, Taiwan, China, January 2006.

[10] C. Chen and W. Lin. *Adaptive Wavelet Sliding Mode Control of Uncertain Multivariable Nonlinear Systems.* In Proceeding of the American Control Conference, pp. 180-185, Denver, Colorado, USA, June 2003.

[11] Y. Cheng, B. Chen, and F. Shiau. *Adaptive Wavelet Network Control Dessign for Nonlinear Systems.* Proceedings of the National Science Counc. ROCA, Vol. 22, num. 6, pp. 783-799, Kobe, Japan, May 1998.

[12] R. Churchill. *Series de Fourier y Problemas de Contorno.* McGraw-Hill, 1978.

[13] R. Crochiere, A. Webber, and J. Flanagan. *Digital Coding of Speech in Subbands.* Bell Systems Technical Journal, Vol. 55, num. 8, pp. 1069-1085, October 1976.

[14] A. Croisier, D. Esteban, and C. Galand. *Perfect Channel Splitting by Use of Interpolation / Decimation / Tree Decomposition Techniques.* In International Conference on Inform. Sciences and Systems, pp. 443-446, Patras, Greece, August 1976.

[15] Carlos Alberto Carrillo Santos, *Controlador PID Wavenet Difuso para Máquinas Eléctricas.* Tesis de maestría. Universisdad Autónoma del Estado de Hidalgo. Mexico. 2009.

[16] Cruz Tolentino J. A., *Diseño y aplicaciones de un controlador PID wavelet.* Tesis de maestría. Universidad Autónoma del Estado de Hidalgo. México. 2009.

[17] Cruz Tolentino J. Alberto, Jarillo Silva Alejandro, Ramos Velasco Luis E. and Domínguez Ramírez Omar A. (2012). *Wavelet PID and Wavenet PID: Theory and Applications, PID Controller Design Approaches - Theory*, Tuning and Application to Frontier Areas, Dr. Marialena Vagia (Ed.), ISBN: 978-953-51-0405-6, InTech.

[18] I. Daubechies. *Ten Lectures on Wavelets.* SIAM, New Jersey, 1992.

[19] I. Daubechies. *The Wavelet Transform, Time-Frequency Localization and Signal Analysis.* IEEE Trans. Inf. Theory, Vol. 36, num. 5, pp. 961-1005, September 1990.

[20] I. Daubechies. *Othonormal Bases of Compactly Supported Wavelets.* Commun. on Pure and Appl. Math., Vol. 41, pp. 909-996, November 1988.

[21] H. Dickhaus and H. Heinrich. *Identification of High Risk Patients in Cardiology by Wavelet Networks.* 18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp.923-924, Amsterdam, Netherlands,1996.

[22] C. R. Domínguez Mayorga, M. A. Espejel Rivera, L. E. Ramos Velasco, J. C. Ramos Fernández y E. Escamilla Hernández, *Algoritmos wavenet con aplicaciones en la aproximación de señales: un estudio comparativo*, Revista Iberoamericana Automática e Informática (RIAI), 2012, ISSN:1697-7912, Vol. 09, pp.347-358, http://dx.doi.org/10.1016/j.riai.2012.09.001

[23] F.A. Díaz López, *Control multiresolución wavenet para sistemas no lineales MIMO: aplicación en una interfaz háptica*, UPP 2013.

[24] F.A. Díaz López, L.E. Ramos Velasco, O. A. Domínguez Ramírez and V. Parra Vega, *Multiresolution Wavenet PID Control for Global Regulation of Robots*, 9th Asian Control Conference (ASCC 2013), June 23-26, 2013 at Istanbul, Turkey.

[25] M. Frezier. *An Introduction to Wavelets Through Linear Algebra.* Springer, 1999.

[26] Fausett L., *Fundamentals of neural networks: Architecture, algorithms and applications*. Prentice Hall, 1994.

[27] R. Feris, J. Kentaro and W. Kruger. *Hierarchical Wavelet Networks Facial Feature Localization.* In proceedings of the 5th IEEE International On Automatic Face and Gesture Recognition, 2002.

[28] J. Fourier. *Théorie Analytique de la Chaleur*. Firmin Didot, Paris, France, 1822.

[29] Freeman J. A., *Simulating neural networks with mathematica*. Reading, MA: Addison-Wesley, 1994.

[30] Gaviphat L., *Adaptive Self-Tuning Neuro Wavelet Network Controllers*. Tesis doctoral. Estados Unidos. 1997.

[31] J. Goswami and A. Chan. *Fundamentals of Wavelets - Theory, Algorithms and Applications*. John Wiley And Sons, 1999.

[32] Graps A., *An introduction to wavelets*. IEEE Computational Science and Engineering, Summer 1995, vol. 2, num. 2.

[33] A. Grossman and J. Morlet. *Decomposition of Hardy Funtions into Squared Integrable Wavelets of Constant Shape*. SIAM J. of Math. Annal., Vol. 15, num. 4, pp. 723-736, July 1984.

[34] A. Haar. *Zur theorie der Orthogonalen Funktionsysteme*. Math. Annal., Vol. 64, pp. 331-373, 1910.

[35] F. Ham and I. Kostanic. *Principles of Neurocomputing of Science and Engineering.* McGraw Hill, New York, USA, 2001.

[36] Hashimoto H., M. Boss, Y. Kuni and F. Harashima, *Intelligent Cooperative Manipulation System Using Dynamic Force Simulator*, Proceedings of IEEE International Conference on Robotics and Automation, IEEE, New York, pp. 2598-2603, May, 1994.

[37] S. Haykin. *Kalman Filtering and Neural Networks.* Wiley, 2001.

[38] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, Upper Saddler River, New Jersey, USA, 1999.

[39] G. Hans, *Wavelets and Signal Processing : An Application-Based Introduction.* Springer, 2005.

[40] Hertz J., Krogh A. and Palmer R.G., *Introduction to the theory of neural computation.* Santa Fe Institute Studies in the Sciences of Complexity (vol. 1). Redwood City, CA: Addision-Wesley. 1991

[41] A. Hojjat, *Wavelets in Intelligent Transportation System.* Wiley, 2004

[42] J. J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities.* Proceedings of the National Academy of Sciences, Vol. 79, pp. 2554-2558, 1982.

[43] H. Karimi, P. Maralami, B. Moshiri and B. Lohmann. *Haar Wavelet-Based Optimal Control of Time-Varying State-Delayed Systems: A Computational Method.* IEEE International Symposium on Computer-Aided Control Systems Design, pp. 1-6, Munich, Germany, Octuber, 2006.

[44] S. Kilts. *ADVANCED FPGA DESIGN. Architecture, Implementation and Optimization.* WILEY, 2007.

[45] T. Kohonen. *Correlation matrix memories.* IEEE Transactions on Computers, Vol. 21, pp. 353-359, 1972.

[46] B. Kuo. *Digital Control Systems.* Oxford University Press, USA, 1995.

[47] F. Keinert. *Wavelets and Multiwavelets.*Chapman and Hall, 2004.

[48] S. Lee. *ADVANCED DIGITAL LOGIC DESIGN. Using verilig, States Machines and Synthesis for FPGAs*, THOMSON, 2006.

[49] C. Lin, Y. Lee, P. Chen and T. Chen. *Multiple Cardiac Arrhytmia Recognition Using Adaptive Wavelet Networks.* In Proceedings of the 2005 IEEE Engineering in Medicine and Biology, 27th Annual Conference, pp. 5655-5659, Shangai, China, September, 2005.

[50] C. K. Lin. *Adaptive TRacking Controller Design for Robotic Systems Using Gaussian Wavelet Networks.* IEEE Proceedings Control Theory Applications, Departement of Electrical Engeneering Chinese Naval Academy of Kaohsiung, Vol 149, num 4, pp. 316-322., China, July 2002.

[51] A. Levin and K. Narendra. *Control of nonlinear dynamical systems using neural networks: Controllability and stabilization.* IEEE Transactions on Neural Networks, 4(2):192 206, March 1993.

[52] Li H., Jin H., and Guo C., *PID Control Based on Wavelet Neural Network Identification and Tuning and Its Application to Fin Stabilizer.* Proceedings of the IEEE International Conference on Mechatronics and Automation, pp. 1907-1911, Niagara Falls, Canada, July 2005.

[53] Lippmann R. P.. *An introduction to computing with neural nets.* IEEE ASSP Magazine , 4 - 22. 1987

[54] S. Mallat. *A Wavelet Tour of Signal Processing.* Prentice Hall PTR, New York, 1999.

[55] S. Mallat. *A Theory Multiresolution Signal Decomposition: The Wavelet Representation.* IEEE Trans. Pattern. Annal. Machine Intell., Vol. 11, num. 7, pp. 674-693, July 1989.

[56] S. Mallat. *Multiresolution Aproximations and Wavelet Orthonormal Bases of $L^2(R)$.* Trans. Amer. Math. Soc., Vol. 315, pp. 69-87, September 1989.

[57] McCulloch W. and Pitts W. *A Logical Calculus of Ideas Imminent in Nervous Activity.* Bull. Math. Biophys, Vol. 5, pp. 115-133, 1943.

[58] McCulloch W. and Pitts W., *How we Know Universal.* Bull. Math. Byops, pp. 127-147, 1947.

[59] Y. Meyer. *Méthodes Temps-Fréquence et Méthodes Temps-Echelle en Traitement du Signal et de L'imagen.* INRIA lectures, 1990.

[60] Y. Meyer. *Ondelettes et Opérateurs.* Hermann, Paris, Francia, 1990.

[61] Y. Meyer. *Wavelets, Algorithms and Applications.* SIAM, Philadelphia, USA, 1993.

[62] H. Moghbelli, A. Rahideh, and A. Safavi. *Vector Control of Induction Machines Using Wavenet Based Controller For Traction Applications.* In IEEE Annual Meeting, Conference record of the 2007, Industry Applications Conference, pp. 761-767, University of Technology of Isfahan, Iran 2003.

[63] Moore, M. and J. Wilhelms, *Collision Detection and Response for Computer Animation*, Computer Graphics, Vol. 22, No. 4, pp. 369-374, 1988.

[64] M. Morris Mano. *Diseño Digital*. PEARSON EDUCATION, tercera edición, 2003

[65] M. Johnson, M. Moradi. *PID Control, New Identification and Design Methods*. Springer-Verlag. Reino Unido. 2005.

[66] X. Ñiu, P. Ching and Y. Chan. *A Wavelet-Based Algorithm for Time Delay and Doppler Measurements.* In International Symposium on Signals, Systems and Electronics, pp. 485-488, San Francisco USA, 1995

[67] K. Ogata. *Discrete-Time Control Systems.* Pretince-Hall, New York, 1995.

[68] K. Ogata. *Ingeniería de Control Moderna.* Pretince-Hall, New York, 1998.

[69] C. Park and P. Tsiotras. *Approximations to Optimal Feedback Control Using a Succesive Wavelet Collocation Algorithm.* In Proceedingsof the American Control Conference, pp.1950-1955, Atlanta Giorgia, USA, June 2003.

[70] S. Parvez and Z. Gao. *A Wavelet-Based Multiresolution PID Controller.* IEEE Transactions on Industry Aplications, Vol. 41, num. 2, pp. 537-543, Hills, Ohio, USA, March - April, 2005.

[71] R. Polikar. *The Wavelet Tutorial*. Dept. of Electrical an Computer Engineering, Rowan University, 1996.

[72] M. Polycarpou, M. Mears, and S. Weaver. *Adaptive Wavelet Control of Nonlinear Systems.* In Proceeding of the 36th Conference on Desicion and Control, pp. 3890-3895, Sandiego, California, USA, December 1997.

[73] J. Proakis. *Introduction to Digital Signal Processing*. Macmillan publising, 1988.

[74] L. E. Ramos Velasco, J.C. Ramos Fernández, O. Islas-Gómez, J. García-Lamont, M.A. Espejel-Rivera, M.A. Márquez Vera, *Identificación y control wavenet de un motor de CA*, Revista Iberoamericana Automática e Informática (RIAI), Elsevier, 2013, ISSN:1697-7912.

[75] Razi M.A. and Athappilly K.. *Expert Systems with Applications* 29 (2005) 65 - 74)

[76] I. Razo. *Aprendizaje Reforzado para el Control de un Sistema Subactuado.* Tesis de Maestría en Ciencias Computacionales, Universidad Autónoma del Estado de Hidalgo, Pachuca, Hidalgo, México, Enero, 2007.

[77] B. Horne, R. Hush. *Progress in Supervise Neural Network. What's New Since Lippmann?.* IEEE Signal Processing Magazine, pp. 8-39, January, 1993.

[78] Rumelhart D.E., Hinton G.E. and Williams R.J., *Learning representations by back-propagating error.* Nature, 323, 533 - 536. Reprinted in Anderson and Rosenfeld [1988], pp. 696 - 699. 1986.

[79] Rumelhat D.E. and L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MA:MIT Press, Vol. 1, 1986.

[80] S. Sanchez-Solano, A.J. Canrera, I. Baturone, F.J. Moreno-Velo, and M.Brox, *FPGA implementation of embedded fuzzy controllers for robotic applications*, IEEE Transactions on Industrial Electonics, vol. 54, no. 4, pp. 1937-1945, 2007.

[81] Shankar Sastry and Marc Bodson, *Adaptive Control: Stability, Convergence, and Robustness*, Prentice-Hall, 1989-1994.

[82] Sedighizadeh and Rezazadeh, *Adaptive PID Control of Wind Energy Conversion Systems Using RASP1 Mother Wavelet Basis Function Networks.* Proceedings of World Academy of Science, engineering and Technology, Vol. 37. 2008.

[83] Y. Sheng. *The Transforms and Applications Handbook.* CRC Press, 1996.

[84] A.D. Stefano, and C. Giaconia, *An FPGA based adaptive fuzzy coprocessor*, Lecture Notes in Computer Science, vol. 3512, pp. 590-597, 2005.

[85] M. Speigel. *Teoría y Problemas de Anáalisis de Fourier.* McGraw-Hill serie de compendios Shaum, 1981.

[86] Y. Tong, Q. Dao, and F. Xu. *AC Motor Control Based on Wavelet Network.* In Proceeding of the Third International Conference on Machine Learning and Cybernetics, pp. 861-865, Shangai, China, August 2004.

[87] M. Vetterli and J. Kovačević. Wavelets and Subband Coding. Prentice Hall PTR, Englewood Cliffs, New Jersey, USA, 2007.

[88] P. Viñuela and I. León, *Redes de Neuronas Artificiales un Enfoque Práctico.* Person Prentice Hall, España, 2004.

[89] Dali Wang, and Ying Bai, *Implementation of fuzzy logic control systems*, Springer, London, 2006, pp. 37-52.

[90] Webster, M., *Websterss Ninth New Collegiate Dictionary*, Merriam-Webster Inc., Springfeld, MA, 1985.

[91] B. Widrow and M. E. Hoff. *Adaptive switching cicuits.* 1960 IRE WESCON Convention Record, New York, Part 4, pp. 96-104, 1960.

[92] J. Xu and Y. Tan. *Nonlinear Adaptive Wavelet Control Using Constructive Wavelet Networks.* In Proceedings of the American Control Conference on Electronics, pp. 624-629, Arlington, USA, June 2001.

[93] Y. Tong, Q. Dao, and F. Xu, *AC Motor Control Based on Wavelet Network.* Proceeding of the Third International Conference on Machine Learning and Cybernetics, pp. 861-865, Shangai, China, August 2004.

[94] S. Zarantonello. *Theory and Application of Wavelets.* Santa Clara University, 1997.

[95] W. Zhou and M. Jamshidi. *Inteligent Control Systems Using Softcomputing Methodologies.* CRC Press,N.Y.,2001.

[96] D. Zhou, W. Cai and W. Zhang. *An Adaptive Wavelet Method for Nonlinear Circuit Simulation.* IEEE Transactions on Circuits and Systems: Fundamental Theory and Applications, Vol 46, num 8, pp. 931-938, August 1999.

[97] A. Zilouchian and M. Jamshidi. *Inteligent Control Systems Using Softcomputing Methodologies.* CRC Press, N. Y., 2001.